Collection-Based Long-Term Preservation

by

Reagan Moore, Chaitan Baru, Amarnath Gupta, Bertram Ludaescher, Richard Marciano, Arcot Rajasekar

San Diego Supercomputer Center
San Diego, California

Submitted to
National Archives and Records Administration

June 1999

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Archives and Records Administration or others supporting the San Diego Supercomputer Center.

Table of Contents

Abstract

The preservation of digital information for long periods of time is becoming feasible through the integration of archival storage technology from supercomputer centers, information models from the digital library community, and preservation models from the archivist's community. The supercomputer centers provide the technology needed to store the immense amounts of digital data that are being created, while the digital library community provides the mechanisms to define the context needed to interpret the data. The coordination of these technologies with preservation and management policies defines the infrastructure for a collection based persistent archive [1]. This report demonstrates the feasibility of maintaining digital data for hundreds of years through detailed prototyping of persistent archives for nine different data collections.

1. Introduction

Supercomputer centers, digital libraries, and archival storage communities have common persistent archival storage requirements. Each of these communities is building software infrastructure to organize and store large collections of data. An emerging common requirement is the ability to maintain data collections for long periods of time. The challenge is to maintain the ability to discover, access, and display digital objects that are stored within the archive, while the technology used to manage the archive evolves. We have implemented an approach based upon the storage of the digital objects that comprise the collection, augmented with the meta-data attributes needed to dynamically recreate the data collection. This approach builds upon the technology needed to support extensible database schema, which in turn enables the creation of data handling systems that interconnect legacy storage systems.

The long-term storage and access of digital information is a major challenge for federal agencies. The rapid change of technology resulting in obsolescence of storage media, coupled with the very large volumes of data (terabytes to petabytes in size) appears to make the problem intractable. The concern is that when the data storage technology becomes obsolete, the time needed to migrate to new technology may exceed the lifetime of the hardware and software systems that are being used. This is exacerbated by the need to be able to retrieve information from the archived data. The organization of the data into collections must also be preserved in the face of rapidly changing technology. Thus each collection must be migrated forward in time onto new management systems, simultaneously with the migration of the individual data objects onto new media. The ultimate goal is to maintain not only the bits associated with the original data, but also the context that permits the data to be interpreted. In this paper we present a scalable architecture for managing media migration, and an information model for managing migration of the structure of the context. For relational databases, the information model includes the schema for organizing attributes and the data dictionary for defining semantics. For hierarchical databases, the information model includes a representation of the hierarchical structure along with the data dictionary.

We rely on the use of collections to define the context to associate with digital data. The context is defined through the creation of hierarchical representations for both the digital objects and the associated data collection. Each digital object is maintained as a tagged structure that includes the original bytes of data, as well as attributes that have been defined as relevant for the data collection. The collection context is defined through use of both hierarchical and relational representations for organizing the collection attributes. By using infrastructure independent representations, the original context for the archived data can be maintained. A collection-based persistent archive is therefore one in which the organization of the collection is archived simultaneously with the digital objects that comprise the collection [1].

A persistent collection requires the ability to dynamically recreate the collection on new technology. For a solution, we consider the integration of scalable archival storage technology from supercomputer centers, infrastructure independent information models from the digital library community, and preservation models from the archivist's community. An infrastructure that supports the continuous migration of both the digital objects and the data collections is needed. Scalable archival storage systems are used to ensure that sufficient resources are available for continual migration of digital objects to new media. The software systems that interpret the infrastructure independent representation for the collections are based upon generic digital library systems, and are migrated explicitly to new platforms. In this system, the original representation of the digital objects and of the collections does not change. The maintenance of the persistent archive is then achieved through application of archivist policies that govern the rate of migration of the objects and the collection instantiation software.

The goal is to preserve digital information for at least 400 years. This report examines the technical issues that must be addressed, evaluates possible implementations, assigns metrics for success, and examines business models for managing a collection-based persistent archive. The applicability of the results is demonstrated by examination of nine different data collections, provided by the USPTO and other federal/state agencies. The report is organized into sections to provide a description of the scaling issues, a generic description of the technology, a comparative synopsis of the nine collections, and detailed descriptions of the approaches that were used for each collection.

2. Technical Issues

The preservation of the context to associate with digital objects is the dominant issue for collection-based persistent archives. The context is traditionally defined through specification of attributes that are associated with each digital object. The context is also defined through the implied relationships that exist between the attributes, and the preferred organization of the attributes in user interfaces for accessing the data collection. We identify three levels of context that must be preserved:

- Digital object representation. For semi-structured data, the organization of the components must be specified, as well as the elements that are used to define the attributes to associate with the collection. An example is a multi-media digital object that has associated text, images, and video. A hierarchical representation is needed to define the relationship between the components, including elements consisting of tagged meta-data attributes.
- Data collection representation. The collection also has an implied organization, which today may be specified through a relational schema or a hierarchical structure. A schema is used to support relational queries of the attributes or meta-data. It is possible to reorganize a collection into multiple tables to improve access by building new indexes, and in the more general case, by adding attributes. The structure used to define the collection attributes can be different from the structure used to specify a digital object within the collection. While relational representations can be used today, in the future, alternate representations may be based upon hierarchical or multi-dimensional algorithms.
- Presentation representation. The user interface to the collection can present an organization of the collection attributes that is tuned to meet the needs of a particular community. Researchers may need access to all of the meta-data attributes, while students are interested in a subset. The structure used to define the user interface again can be different from the schema used for the collection organization. Each of these presentations represents a different view of the collection. Re-creation of the original view of a collection may or may not be possible.

Digital objects are used to encapsulate each data set. Collections are used to organize the context for the digital objects. Presentation interfaces are the structure through which collection interactions are defined. The challenge is to preserve all three levels of context for each collection.

2.1 Managing Context

Management of the collection context is made difficult by the rapid change of technology. Software systems used to manage collections are changing on five to ten-year time scale. It is possible to make a copy of a database through a vendor specific dump or backup routine. The copy can then be written into an archive for long term storage. This approach fails when the database is retrieved from storage, as the database software may no longer exist. The archivist is then faced with migrating the data collection onto a new database system. Since this can happen for every data collection, the archivist will have to continually transform the entire archive. A better approach is needed.

An infrastructure independent representation is required for the collection that can be maintained for the life of the collection. If possible, a common information model should be used to define the hierarchical structures associated with the digital objects, the collection organization, and the presentation interface. An emerging standard is the eXtended Markup Language (XML) [2]. XML provides an information model for describing hierarchical structures through use of nested elements. Elements are essentially tagged pieces of data. A Document Type Definition (DTD) provides the particular hierarchical organization that is associated with a given document or digital object. XML Style Sheet Language (XSL) can be used to define the presentation style to associate with a DTD. It is possible to use multiple style sheets for a given DTD. This provides the flexibility needed to represent the context for the user interface into a collection, as well as the structure of the digital objects within the collection.

Although DTDs were originally applied to documents, they are now being applied to arbitrary digital objects, including the collections themselves. XML DTDs can be used to define the structure of digital objects, specify inheritance properties of digital objects, and define the collection organization and user interface structure. DTDs can also be used to define the structure of highly regular data or semi-structured data. Thus DTDs are a strong candidate for a uniform information model.

While XML DTDs provide a tagged structure for organizing information, the semantic meaning of the tags used within a DTD is arbitrary, and depends upon the collection. A data dictionary is needed for each collection to define the semantics. A persistent collection therefore needs the following components to define the context:

- Data dictionary for collection semantics,
- DTD for digital object hierarchical structure,
- DTD for collection hierarchical structure,
- DTD for user interface structure,
- XSL style sheets for presentation of each DTD.

2.2 Managing Persistence

Persistence is achieved by providing the ability to dynamically reconstruct a data collection on new technology. While the software tools that do the reconstruction have to be ported to work with each new hardware platform or database, the collection can remain in its original format within an archive. The choice of the appropriate standard for the information model is vital for minimizing the support requirements for a collection-based persistent archive. The goal is to store the digital objects comprising the collection and the collection context in an archive a single time. This is possible if any changes to the information model standard contain a superset of the prior information model. The knowledge required to manipulate a prior version of the information model can then be encapsulated in the software system that is used to reconstruct the collection.

With this caveat, the persistent collection never needs to be modified, and can be held as infrastructure independent bit-files in an archive.

The re-creation or instantiation of the data collection is done with a software program that uses the DTDs that define the digital object and collection structure to generate the collection. While the current prototypes rely on a different collection instantiation program for each collection, the goal is to build a generic program that works with any DTD. This will reduce the effort required to support dynamic reconstruction of a persistent data collection to the maintenance of a single software system.

Maintaining persistent digital objects requires the ability to migrate data to new media. The reasons for continuing to refresh the media on which the collection is maintained are:

- Avoid loss of data because of the finite lifetime and resulting degradation of the media.
- Minimize storage costs. New media typically store at least twice as much data as the prior version, usually at the same cost per cartridge. Thus migration to new media results in the need for half as many cartridges, decreased floor space, and decreased operating costs for managing the cartridges. Note that for this scenario, the media costs for a continued migration will remain bounded, and will be less than twice the original media cost. The dominate cost to support a continued migration onto new media is the operational support needed to handle the media.
- Maximize the ability to handle exponentially increasing data growth. Many data collections are doubling in size in time periods shorter than a year. This means the effort to read the entire collection for migration to new media will be less than the effort to store the new data that is being collected within that year. Migration to higher density media is the only way to keep the number of cartridges to a manageable level.

To facilitate migration and access, supercomputer centers keep all data in tape robots. For currently available tape (cartridges holding 20 GB to 50 GB of data), a single tape robot is able to store 120 terabytes to 300 terabytes of uncompressed data. By year 2003, a single tape robot is expected to hold 6000 terabytes, using 1-terabyte capacity cartridges. The storage of petabytes (thousands of terabytes) of data is now feasible.

Given that the collection context and the digital objects can be migrated to new media, the remaining system that must be migrated is the archival storage system itself. The software that controls the tape archive is composed of databases to store the storage location and name of each data set, logging systems to track the completion of transactions, and bitfile movers for accessing the storage peripherals. Of these components, the most critical resource is the database or nameserver directory that is used to manage the names and locations of the data sets. At the San Diego Supercomputer, the migration of the nameserver directory to a new system has been done twice, from the DataTree archival storage system to the UniTree archival storage system, and from UniTree to the IBM High Performance Storage System [3]. Each migration required the read of the old directory, and the ingestion of each data set into the new system. In Table 2-1, the times and number of data sets migrated are listed. Note that even though the number of files dramatically increases, the time required for the migration decreased. This reflects advances in vendor supplied systems for managing the name space. Based on this experience, it is possible to migrate to new archival storage systems, without loss of data.

System Migration	Number of files	Time (days)
DataTree to UniTree	4 million	4
UniTree to HPSS	7 million	1

Table 2-1. Migration of archival storage system nameserver directory

One advantage of archival storage systems is their ability to manage the data movement independently from the use of the data. Each time the archival storage system was upgraded, the new version of the archive was built with a driver that allowed tapes to be read from the old system. Thus migration of data between the archival storage systems could be combined with migration onto new media, minimizing the number of times a tape had to be read.

The creation of a persistent collection can be viewed as the design of a system that supports the independent migration of each internal hardware and software component to new technology. Management of the migration process then becomes one of the major tasks for the archivist.

2.3 Managing Scalability

A persistent archive can be expected to increase in size through either addition of new collections, or extensions to existing collections. Hence the architecture must be scalable, supporting growth in the total amount of archived data, the number of archived data sets, the number of digital objects, the number of collections, and the number of accesses per day. These requirements are similar to the demands that are placed on supercomputer center archival storage systems. We propose a scalable solution that uses supercomputer technology, based on the use of parallel applications running on parallel computers.

Archival storage systems are used to manage the storage media and the migration to new media. Database management systems are used to manage the collections. Web servers are used to manage access to the system. A scalable system is built by identifying both the capabilities that are best provided by each component, and the constraints that are implicit within each technology. Interfaces are then constructed between the components to match the data flow through the architecture to the available capabilities. Table 2-2 lists the major constraints that the architecture must manage for a scalable system to be possible.

Archival storage systems excel at storing large amounts of data on tape, but at the cost of relatively slow access times. The time to retrieve a tape from within a tape silo, mount the tape into a tape drive, and ready the tape for reading is on the order of 15-20 seconds for current tape silos. The time required to spin the tape forward to the position of the desired file is on the order of 1-2 minutes. The total time can be doubled if the tape drive is already in use. Thus the access time to data on tape can be 2-4 minutes. To overcome this high latency, data is transferred in large blocks, such that the time it takes to transfer the data set over a communication channel is comparable to the access latency time. For current tape peripherals which read at rates from 10 MB/sec to 15 MB/sec, the average data set size in an archive should be on the order of 500 MB to 1 GB. Since digital objects can be of arbitrary size, containers are used to aggregate digital objects before storage into the archive.

Component	Capability	Constraints
Archive	Massive storage	Latency of access Number of data sets
Database	Large number of objects	Access optimization Query language
Web Server	Ubiquitous access	Presentation format Storage capacity

Table 2-2. Architecture Components

The second constraint that must be managed for archives is the minimization of the number of data sets that are seen by the archive. Current archival storage nameservers are able to manage on the order of 10 – 40 million data sets. If each data set size is on the order of 500 MB, the archive can manage about 10 petabytes of data (10,000 TBs, or 10 million GBs). Archival storage systems provide a scalable solution only if containers are used to aggregate digital objects into large data sets. The total number of digital objects that can be managed is on the order of 40 billion, if one thousand digital objects are aggregated into each container.

Databases excel at supporting large numbers of records. Note that the Transaction Processing Council D benchmark [4] measures performance of relational databases on decision support queries for database sizes ranging from 1 gigabyte up to 3 terabytes and from 6 million to 18 billion rows. Each row can represent a separate digital object. With object relational database systems, a binary large object or BLOB can be associated with each row. The BLOBs can reside either internally within the database, or within an external file system. In the latter case, handles are used to point to the location of BLOB. The use of handles makes it feasible to aggregate digital objects within containers. Multiple types of container technology are available for aggregating digital objects. Aggregation can be done at the file level, using utilities such as the TAR program, at the database level through database tablespaces, or at an intermediate data handling level through use of software caches. All three approaches are demonstrated in the persistent collection prototyping efforts described in section 4. The database maintains the information needed to describe each object, as well as the location of the object within a container and the location of the container within the storage system. A data handling system is used to support database access to archival storage.

Queries are done across the attributes stored within each record. The time needed to respond to a query is optimized by constructing indexes across the database tables. This can reduce the time needed to do a query by a factor of a thousand, at the cost of the storage space for the index, and the time spent in assembling the index. Persistent collections may be maintained on disk to support interactive access, or they may be stored in the archive, and rebuilt on disk when a need arises. If the collection is reassembled from out of the archive, the dominant time needed for the process may be the time spent creating a new index. Since archival storage space is cheap, it may be preferable to keep both infrastructure independent and infrastructure dependent representations of a collection. The time needed to load a pre-indexed database snapshot is a small fraction of the time that it would take to reassemble and index a collection. The database snapshot, of course, assumes that the database software technology is still available for interpreting the database snapshot. For data collections that are frequently accessed, the database snapshot may be worth maintaining.

The presentation of information for frequently accessed collections requires Web servers to handle the user load. Servers function well for data sets that are stored on local disk. In order to access data that reside within an archive, a data handling system is needed to transfer data from the archive to the Web server. Otherwise the size of the accessible collection may be limited to the size of the Web server disk cache. Web servers are available that distribute their load across multiple CPUs of a parallel computer, with parallel servers managing over 10 million accesses per day.

Web servers provide a variety of user interfaces to support queries and information discovery. The preservation of the user interface requires a way to capture an infrastructure independent representation for the query construction and information presentation. Web servers are available that retrieve information from databases for presentation. What is needed is the software that provides the ability to reconstruct the original view of the collection, based upon a description of the collection attributes. Such technology is demonstrated as part of the collection instantiation process.

2.4 Managing Heterogeneity of Data Resources

A persistent archive is inherently composed of heterogeneous resources. As technology evolves, both old and new versions of the software and hardware infrastructure will be present at the same time. An issue that must be managed is the ability to access data that is present on multiple storage systems, each with possibly different access protocols. A variant of this requirement is the ability to access data within an archive from a database that may expect data to reside on a local disk file system. Data handling systems provide the ability to interconnect archives with databases and with Web servers. Thus the more general form of the persistent archive architecture uses a data handling system to tie each component together. At the San Diego Supercomputer Center, a particular implementation of a data handling system has been developed, called the Storage

Resource Broker (SRB) [5]. A picture of the SRB architecture is shown in Figure 2-1 to illustrate the required components.

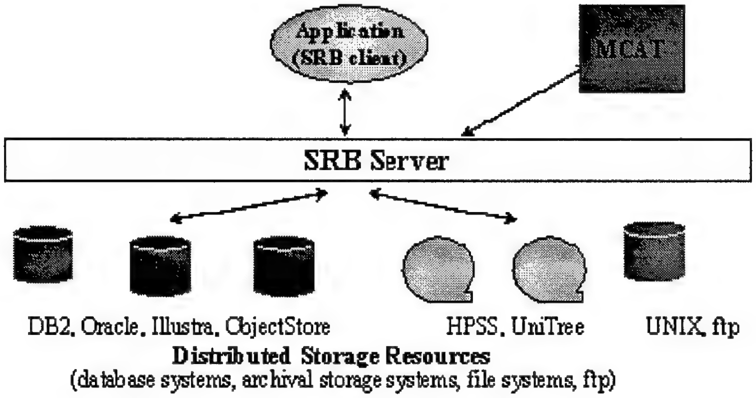

Application (SRB client)
MCAT
SRB Server
DB2, Oracle, Illustra, ObjectStore
HPSS, UniTree
UNIX, ftp
Distributed Storage Resources
(database systems, archival storage systems, file systems, ftp)

Figure 2-1. SDSC Storage Resource Broker Architecture

The SRB supports the protocol conversion needed for an application to access data within either a database, file system, or archive. The heterogeneous nature of the data storage systems is hidden by the uniform access API provided by the SRB. This makes it possible for any component of the architecture to be modified, whether archive, or database, or Web server. The SRB Server uses a different driver for each type of storage resource. The information for which driver to use for access to a particular data set is maintained in the associated Meta-data Catalog (MCAT) [6,7]. The MCAT system is a database containing information about each data set that is stored in the data storage systems. New versions of a storage system are accessed by a new driver written for the SRB. Thus the application is able to use a persistent interface, even while the storage technology changes over time.

3. Implementation strategy

A collection based persistent archive can be assembled using a scalable architecture. The scalable architecture relies upon parallel hardware and software technology that is commercially available. The persistent archive requires the integration of three separate components; archival storage, collection management, and access servers through the use of a data handling system. The result is a system that can be modified to build upon new technology on an incremental basis. For a persistent archive to work within this migration environment, the data context must be maintained in an information independent representation. The technology to instantiate the collection will have to be migrated forward in time, along with the data handling system. The collection can be kept as bit-files within the archive, while the supporting hardware and software systems evolve.

3.1 General Architecture

The implementation of a persistent archive at SDSC is based upon use of commercially available software systems, augmented by application level software developed at the San Diego Supercomputer Center. The architecture software components are:

- Archival storage system – IBM High Performance Storage System (HPSS) [3]
- Data handling system – SDSC Storage Resource Broker (SRB) [5]
- Object relational database – Oracle version 7.3, IBM DB2 Universal Database
- Collection management software – SDSC Meta-data Catalog (MCAT) [6,7]
- Collection instantiation software – SDSC scripts
- Collection ingestion software – SDSC scripts
- Hierarchical data model – eXtended Markup Language – Document Type Definition [2]
- Relational data model – ANSI SQL Data Definition Language [8]
- DTD manipulation software – UCSD XML Matching and Structuring language (XMAS) [9]
- Web server – Apache Web server
- Presentation system – Internet Explorer version 5

The hardware components are:

- Archival storage system – IBM SP 8-node, 32-processor parallel computer, 180 TB of tape storage, three Storage Technology tape robots, and 1.6 TB of RAID disk cache
- Data management system – Sun Enterprise 4-processor parallel computer
- Data ingestion platform – SGI workstation
- Network interconnect – Ethernet, FDDI, and HiPPI

Each of these systems is scalable, and can be implemented using parallel computing technology. The efficiency of the archival storage system is critically dependent upon the use of containers for aggregating data before storage. Three difference mechanisms have been tried at SDSC:

- Unix utilities. The TAR utility can be used to aggregate files. For container sizes of 100 MB, the additional disk space required is minimal. The disadvantages are that the container must be read from the archive and unpacked before data sets are accessed.
- Database tablespace. At SDSC, a prototype version of the DB2 UDB [10] parallel object-relational database has been used to support large data collections. The prototype database stores the digital objects internally within tablespaces. The tablespaces can be stored within the HPSS archival storage system, and retrieved to a disk cache on demand. This effectively increases the database storage capacity to the size of the archive, while simultaneously aggregating digital objects into containers before storage in the archive.
- Data handling software cache. The SDSC Storage Resource Broker supports containers. Digital objects that are written into an archive through the SRB are aggregated into a container on a disk cache. When the container is full, the SRB writes the container into the archive. When data is referenced, the container is retrieved from the archive and the data set is read directly out of the container by the SRB.

3.1.1 Archive

The core of the architecture is the archival storage system, as it ultimately determines the total capacity, data ingestion rate, and data migration support for the persistent archive. The architecture of the HPSS [2,11] archive is shown in Figure 3-1. The system at SDSC currently stores over 9 million files, with an aggregate size of 100 TB. Data movement rates have been achieved that exceed 2 TBs of data storage per day. The system sustains 16,000 file operations per day, with the ability to handle over 100,000 file operations per day. The HPSS system is accessed over high-speed networks through a High Performance Gateway Node (HPGN). The HPGN supports multiple types of network access, including a 100 MB/sec HiPPI network, 100 Mb/sec FDDI, and Ethernet. The HPGN is directly connected to the nodes of the SP on which the HPSS software system runs through the Trail Blazer 3 switch. The HPSS central control services run on one of the four-processor nodes, while the bitfile movers that read/write data off of disk and tape are distributed across seven of the SP nodes. By interconnecting the external networks through the HPGN onto the SP switch, all of the mover nodes can be used in parallel, sustaining high data throughput. By having disk and tape drives connected to each of the mover nodes, data can be migrated in parallel to tape. Measured data movement rates from the nodes to the HPGN are 90 MB/s for file sizes on the order of 10 MB.

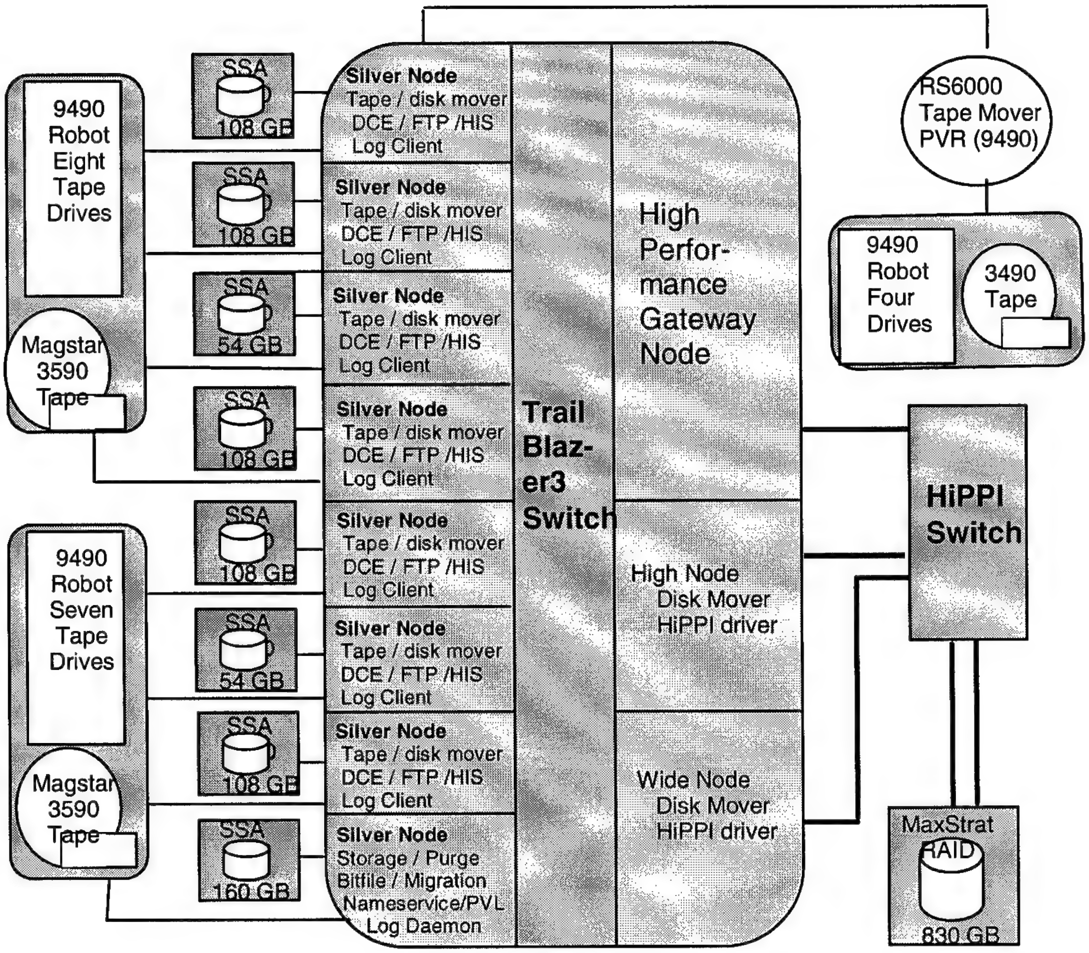


Figure 3-1. HPSS archival storage system

The system includes multiple backup systems for preserving the nameserver directory, including mirroring of the directory on disk, backup of snapshots of the directory onto tape, transaction logging of all changes to the directory, and reconciliation of the transaction logs with the directory snapshots on a daily basis. To handle disasters, copies of the critical data sets are maintained in a second HPSS archival storage system located within another city. A description of the backup systems is given in [12]. The attention paid to nameserver directory backup is of critical importance. If the nameserver directory is lost, it will not be possible to provide the names for the files stored in the archive.

The system is scalable, through the addition of more nodes, disk, and tape drives. The system will be upgraded to a capacity of 360 GB of uncompressed data in 1999 through the acquisition of tape drives that write 20 GBs of data per cartridge. The system supports data compression. For the scientific data sets stored at SDSC, the average compression ratio is a factor of 1.5, implying the total capacity of the system will be 500 TB.

3.1.2 Data Handling System

The data handling system provides the ability to connect heterogeneous systems together. We provide a detailed description of the SDSC data handling system to illustrate the software infrastructure needed to provide location and protocol transparency. The data handling infrastructure developed at SDSC has two components: the SDSC Storage Resource Broker (SRB) [5] that provides federation and access to distributed and diverse storage resources in a heterogeneous computing environment, and the Meta-data Catalog (MCAT) [6] that holds systemic and application or domain-dependent meta-data about the resources and data sets (and users) that are being brokered by the SRB. The SRB-MCAT system provides the following capabilities:

- uniform APIs for access to heterogeneous file systems, databases, and archival storage,
- protocol-transparency and location-transparency when accessing distributed systems,
- uniform name space abstraction over the file systems that are being brokered,
- meta-data-based access to files, thus supporting information discovery based on domain and system-dependent meta-information stored along with (or extracted from) the stored files,
- facilities for replication, copying or moving files across heterogeneous systems, performing resource-level operations (proxy operations) on data before delivery to the client, and
- an integrated encryption and authentication system that can range from no security to fully encrypted and fully authenticated data transfer including security against man-in-the-middle security intrusions [13,14].

The SDSC Storage Resource Broker (SRB) is middleware that provides distributed clients with uniform access to diverse storage resources in a heterogeneous computing environment. Storage systems handled by the current release of the SDSC SRB include the UNIX file system, archival storage systems such as UniTree, ADSM and HPSS, and database Large Objects managed by various DBMSs including DB2, Oracle, and Illustra (Figure 2-1). Currently, the system runs on supercomputers such as the CRAY C90, CRAY T3E and IBM SP, and on workstations such as Sun, SGI, and DEC platforms. The SRB API presents clients with a logical view of data sets stored in the SRB. Similar to the file name in the file system paradigm, each data set stored in SRB has a logical name, which may be used as a handle for data operation. Unlike the file system where the physical location of a file is implied in its path name through its mount point, the physical location of a data set in the SRB environment is logically mapped to the data sets. Therefore, data sets belonging to the same collection may physically reside in different storage systems. A client does not need to remember the physical mapping of a data set. It is stored as the meta-data associated with the data set in the MCAT catalog. Data sets in the SRB are grouped into a logical (hierarchical) structure called *collections*. The collection provides an abstraction for:

- placing similar objects (possibly, physically distributed) under one collection (e.g., image collections of a museum) and
- placing all dissimilar objects that have a common connection under one abstraction (e.g., all the text paragraphs, images, figures, and tables of a document).

The SRB supports data replication in two ways. One can replicate an object during object creation or modification. To enable this, SRB and MCAT allow the creation of *logical storage resources* (LSR) which are a grouping of two or more resources. When an application creates or writes a data set in these logical resources, then the operations are performed on all the grouped resources. The result of using a LSR is that a copy of the data is created in each of the physical resources belonging to the logical resource. It is possible to specify that the write operation is successful if *k* of the *n* copies are created. The user can modify all the copies of the data by writing to the data set with a "write all." However, this operation can lead to an inconsistency if there is a failure in the middle of the operation. The SRB also provides an off-line replication facility to replicate an existing data set. This operation can also be used for synchronization purposes. When accessing replicated objects, SRB will open the first available replica of the object as given by a list from MCAT. The SRB also provides authentication and encryption facilities [13,14], access control list and ticket-based access [15], and auditing capabilities to give a feature-rich environment for sharing distributed data collections among users and groups of users.

The design of the SRB server is based on the traditional network connected client/server model but has the additional capability of federation. Once a connection from a client is established and authenticated, a SRB agent is created that brokers all the operations for that connection. A client application can have more than one connection to a SRB server and to as many servers as required. The federation of SRBs implies that a client connects to any SRB server while accessing a resource that is brokered by another server. An inter-SRB communication protocol supports the federation operation. The SRB communicates with MCAT to obtain meta-information about the data set, which it then uses for accessing the data set.

3.1.3 Collection Management

Current information models include relational representations of data collections, such as the Data Definition Language, DDL [8]. Relational representations are used by relational databases to define how queries can be decomposed across the multiple tables that are used to hold the meta-data attributes. It is possible to generate arbitrary mappings between a DTD hierarchical representation, and a DDL relational representation for a collection. A preferred correspondence between the two representations must be defined if a relational database is used to assemble the collection.

In section 3.2.2 we describe the process we used to dynamically create a collection. The demonstration is based upon digital library technology that has been developed at SDSC.

While the current technology builds upon object relational databases, in the future it may be possible to work directly with hierarchical databases such as Excelon (an XML variant of ObjectStore) and Ariel (an XML version of O2). This would avoid the need to map between hierarchical and relational schemas.

A detailed description of the SDSC MCAT system is provided to illustrate the complexity of the information management software needed to describe and manage collection level meta-data. The SDSC MCAT is a database catalog that provides a repository of meta information about digital objects. Digital object attributes are separated into two classes of information within the MCAT:

- System-level meta-data that provides operational information. These include information about resources (e.g., archival systems, database systems, etc. and their capabilities, protocols, etc) and data objects (e.g., their formats or types, replication information, location, collection information, etc.).
- Application-dependent meta-data that provides information specific to particular data sets and their collections (e.g., Dublin Core [16,17] values for text objects).

Both of these types of meta-data are extensible, i.e., one can add and/or remove attributes. Internally, MCAT keeps schema-level meta-data about all of the attributes that are defined. The schema-level attributes are used to define the context for a collection and enable the instantiation of the collection on new technology. The attributes include definition of:

- Logical Structure: When a set of meta-data is registered with MCAT, one needs to identify a logical structure in which the rest of the meta-data will be organized. The logical structure should not be confused with database schema and are more general than that. For example, we have implemented the Dublin Core database schema [16] to organize attributes about digitized text. The attributes defined in the logical structure that is associated with the Dublin Core schema contains information about the subject, constraints, and presentation formats that are needed to display the schema along with information about its use and ownership.
- Attribute Clusters: An attribute cluster is a set of attribute names that are logically interconnected and that have a one-to-one mapping among them. One can view them as a (single or a set of) normalized table(s) in a database context. For example, in the Dublin Core, publisher, name, address, and contact information form a cluster. Contributor name and contributor type form a second cluster; title and its type form yet another cluster, and so on. Similarly in our system-level MCAT core meta-data, we have one cluster for each data replica containing the type, location, and size of the data objects. This aids the implementation of relational joins across the meta-data tables, since each replica has only one value for these properties and these properties provide the physical characteristics of the object. For each cluster, MCAT keeps information about any constraints and comments that can be searched when using the attribute, along with information about use-privileges and grant-of-use-privileges for the cluster. For each attribute,

MCAT keeps more than 20 different types of information including its physical, logical and input and output characteristics [9].

- Token Attributes: Token attributes have a specific function (compared to other attributes); they capture some simple semantic information about the domain of discourse. In the simplest sense, one can use the token attributes to provide the *domain of discourse* for an attribute or a set. One can also use the token attribute to capture semantic translation between discipline domains (e.g., common names vs. scientific names) and also capture hierarchical and equivalence relationships in the domain of discourse. Given the development of semantic standards within a discipline, one can use the token attribute as a bridge between two schemas and provide semantic interoperability.
- Linkages: Linkages provide a means for inter-operating within and between schema. One can define four types of linkages:
    1. attribute-to-attribute,
    2. cluster-to-attribute,
    3. cluster-to-cluster, and
    4. cluster-to-token.

  Each of the linkages can be from one-to-many, many-to-one, or many-to-many. The linkage information is used to generate joins dynamically based on the user's chosen set of attributes. The join algorithm uses Steiner Tree generation of SQL commands from a directed acyclic graph; the DAG is a mapping of clusters and the linkages between them. The linkage information is used to perform federated query operations across schemas. The DAG is also used to figure out the notion of an allowed query by disallowing queries that span disjointed graphs.

MCAT provides APIs for creating, modifying and deleting the above structures. The architecture of the MCAT is given in Figure 3-2. MCAT provides an interface protocol for applications such as Web servers. The protocol uses a data structure for the information interchange which is called MAPS—Meta-data Attribute Presentation Structure. The data structure, which also has a wire-format for communication and a data format for computation, provides an extensible model for communicating meta-data information. A mapping is being developed to translate from the MAPS structure to the Z39.50 format [18]. Internal to MCAT, the schema for storing meta-data (may possibly) differ from MAPS, and hence mappings between the internal format and MAPS are needed for every type of implementation of the MCAT. Note that it is possible to store the meta-data in databases, flat files, or LDAP directories [19]. MAPS provides a uniform structure for communicating between MCAT servers and user applications.

The MAPS structure defines a query format, an update format and an answer format. The MAPS query format is used by MCAT in generating joins across attributes based on the schema, cluster and linkages discussed above. Depending upon the internal catalog type (e.g., DB2 database, Oracle database, or LDAP) a lower-level target query is generated. Moreover, if the query spans several database resources, a distributed query plan is generated.

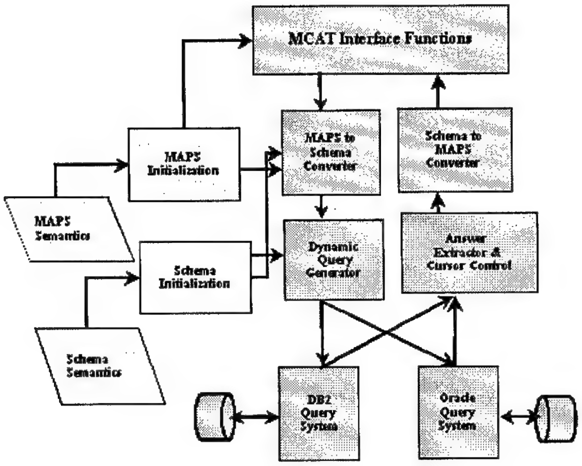


Figure 3-2. MCAT architecture.

The MCAT system supports the publication of schemata associated with data collections, schema extension through the addition or deletion of new attributes, and the dynamic generation of the SQL that corresponds to joins across combinations of attributes. GUIs have been created that allow a user to specify a query by selecting the desired attributes. The MCAT system then dynamically constructs the SQL needed to process the query. By adding routines to access the schema-level meta-data from an archive, it is possible to build a collection-based persistent archive. As technology evolves and the software infrastructure is replaced, the MCAT system can support the migration of the collection to the new technology. Effectively, the collection is completely represented by the set of digital objects stored within the archive, the schema that contains the digital object meta-data, and the schema-level meta-data that allows the collection to be instantiated from scratch.

3.2 Persistent Archive Assessment

The assessment of a particular implementation of a persistent archive is based upon tradeoffs between cost, performance, and risk. Cost is driven by the performance requirements and the acceptable degree of risk. Risk is mitigated by increasing the level of hardware resources that must be dedicated to achieve the desired performance. Archival storage systems typically operate with minimal risk, as long as there are sufficient resources to meet the demand for disk space, CPU cycles, or network bandwidth. When any of these resources becomes scarce, the system will stop functioning. An assessment can be quantified by specifying the minimal levels of resources needed for a given load. In reference [12], this assessment has been done for the HPSS archival storage system. A scaling study has been done to define how the critical resources must be increased in size as the total number of files and the access transaction rate increases. The controlling parameter is the size of the container used to aggregate data for storage into the archive. This impacts the transaction rate, the utilization efficiency for the tape drives, and the effective bandwidth that the system is able to sustain. Table 3-1 gives the amount of directory disk space, the CPU capacity, and the storage needed to log transactions, as a function of the demands on the system. Similar assessments can be done for the database that is used to manage the collection, and are also listed in Table 3-1. The database scaling factors are based upon industry standards [4].

Function	Resource	Scaling
HPSS		
Logging space	Disk	100 MB per 1000 file transactions
Directory space	Disk	2.5 GB per million containers
Data movement	# Nodes	3 * (I/O rate)/(node I/O rate)
Data movement	# Tapes	7 * (I/O rate)/(tape I/O rate)
Data cache	Disk	3 * (daily stored data)
Database		
Directory size	Disk	3 * (meta-data sizeper object) * (# objects)
Request processing	# Nodes	(Directory size) / (30 GB per node)

Table 3-1. Scaling parameters for archive and database size

The scaling study must be augmented with an analysis of theuser load to properly tune the archival storage system. An example analysis for the SDSC user workload is published in reference [20]. The archival storage resources (disk space and tape space) are assigned to service classes that are matched to the load requirements. This guarantees that performance requirements can be met by the archive.

3.2.1 Usage Models

The scaling parameters listed in Table 3-1 assume that the transaction rate for the archive remains below the maximum sustainable transaction rate for the system. For the SDSC storage system, a conservative estimate for the maximum file operation rate is about 100,000 file operations per day. For the E-mail collection described in section 5, the average message size is 2.5 kB. For a 100-MB sized container, it is possible to store 40,000 messages per container. The maximum rate that messages can be loaded into the archive through use of containers is then over 4 billion messages per day.

The controlling parameter then becomes the sustainable I/O rate. The required I/O rate is aggregated across multiple peripherals. For accesses to disk, the internal I/O rate can be a factor of three higher than the external data movement into the archive, counting data migration to tape and caching from tape. For access to tape, the duty cycle of the tape drive must be included, as part of the time a tape drive will be idle while a new tape is loaded and positioned. For 100 MB containers, the duty factor can be as high as 7, if storage of each container requires that a new tape be mounted. The data cache needed for an archive to manage the data movement onto tape is scaled in size to allow three days of data to accumulate. This means the data for the previous day can be processed while the next set of data is being ingested, and other data collections are being queried. The scaling metrics can be evaluated to show that the resources needed for a single data collection are small compared to the archive at the San Diego Supercomputer Center. However, when the number of collections becomes sufficiently large, on the order of 100, the persistent archive requirements approach that of a supercomputer center. We demonstrate this by evaluating the resource requirements for three different types of collections (E-mail, text, and images).

We consider a usage scenario in which data is ingested onto disk, mined for meta-data, aggregated into containers, and stored into the archive. The meta-data is loaded in parallel into a database table for supporting future queries against the collection. For each of the collections, we assume the data is ingested on a daily basis, and calculate the number of CPUs, amount of disk, and number of tape drives that are needed to manage a data collection that has been aggregated over a year. . Note that the database is read from the archive before each daily update, and then written back to the archive. The node disk I/O rate is assumed to be 30 MB/sec, and the tape I/O rate is assumed to be 10 MB/sec. The entire database of meta-data is read and written each day from the archive. The digital objects are written daily into the archive but not re-read. The archive disk is made large enough to support both the data ingestion, and the reading of the database.

An E-mail collection that aggregates 36 million messages per year is shown in Table 3-2.

Load	Rate	Size
	100,000 messages per day	2.5 kB per message
	250 MB of data per day	
	10 MB of meta-data per day	100 B of meta-data per message
	3 containers stored per day	40,000 messages per container
	36 million messages stored per year	
	91 GB of data stored per year	
	3.6 GB of meta-data stored per year	
	900 containers stored per year	100 MB per container

Table 3-2. E-mail collection of 36 million messages per year

The corresponding resource requirements are:

- Archive – 1 node, 12 GB disk, 1 tape drive
- Database – 1 node, 11 GB disk

Load	Rate	Size
	100,000 files per day	30 kB per file
	3 GB of data per day	
	100 MB of meta-data per day	1 kB of meta-data per file

	33 containers per day	3300 files per container
	36 million files per year	
	1.2 TB of data per year	
	36 GB of meta-data per year	
	12,000 containers per year	100 MB per container

Table 3-3. Text collection of 100,000 files per day

Note that current database technology manages up to three billion records on parallel computers.

The same analysis is shown for a text collection in Table 3-3. The corresponding resource requirements are:

- Archive – 1 node, 130 GB disk, 2 tape drives
- Database – 4 nodes, 120 GB disk

Load	Rate	Size
	10,000 images per day	2 MB per image
	20 GB of data per day	
	20 MB of meta-data per day	2 kB of meta-data per image
	200 containers per day	50 images per container
	3.6 million images per year	
	7.3 TB of data per year	
	7.3 GB of meta-data per year	
	73,000 containers per year	100 MB per container

Table 3-4. Image collection of 10,000 images per day

The analysis for an image data collection is shown in Table 3-4. The corresponding resource requirements are:

- Archive – 1 node, 82 GB disk, 1 tape drive
- Database – 1 node, 22 GB disk

While these resource requirements are all more than a factor of 10 smaller than the supercomputer center archive, the requirements become substantial if 100 collections must be managed, each of which is ingesting data on a daily basis. Note that in this usage scenario, each database is read from the archive, meta-data is added to the database and the database is re-written into the archive on a daily basis. One could support the collections by adding resources proportionally to sustain the aggregate I/O rate. This is possible because the total number of files that the archive must manage is governed by the container size, and is only 1 million for a total storage of 100 TB. At this rate the archive could store data for 40 years before the maximum number of containers that can be managed by the archive is exceeded. The archive is scaled in size to handle the increased load by adding CPU, disk and tape resources.

There are several approaches for decreasing the total amount of hardware resources that would be needed. In practice, for ingestion of continuous streams of data, the amount of data that is stored within one collection can be decreased by aggregating data for a shorter period of time, on a bi-annual or monthly basis. This implies that the amount of data that is read on a daily basis to modify the database for each collection can be decreased by a factor of ten if each database contain a month's worth of data. Queries for access would then use a finding aid to locate the correct database for information retrieval.

The usage scenario for data ingestion can also be modified to decrease the amount of I/O by using database technology that maintains unused partitions of its index space in the archive. This is an area of research that could be pursued for integrated database/archival storage systems. Each database would keep a subset of the meta-data continuously on disk, and page meta-data to the archive as meta-data containers fill up.

The capital cost of a persistent archive is proportional to the amount of hardware resources that are required. Table 3-5 gives the requirements for a system that could sustain 100 GB of data ingestion per day, hold 3 years of data (109 TB), manage 1.1 million containers, and manage yearly aggregation of 3.3 million digital objects for each of 100 collections.

Load	Rate	Size
	900,000 digital objects per day	110 kB per digital object
	100 GB of data per day	
	900 MB of meta-data per day	1 kB of meta-data per digital object
	1000 containers per day	900 digital objects per container
	328 million digital objects per year	
	36 TB of data per year	
	328 GB of meta-data per year	
	365,000 containers per year	100 MB per container

Table 3-5. Large scale persistent archive

The archive nodes are assumed to have 4 processors per node, while the database nodes have 1 processor per node. Note that the collections are cycled in turn through the database nodes for update on a daily basis. Thus while the total amount of database directories that are read each day is 1 TB, only 10 GB of meta-data is manipulated at any one time within a database. While one database is being manipulated, the next one is being read and the prior database is being written back to the archive.
Table 3-6 gives the associated resources and approximate cost for the persistent archive that should be capable of sustaining the aggregate data load.

Resource	Number of units	Cost per unit
Archive tape robot	1	$300,000
Archive tape drives	16 drives	$30,000
Tape media (20 GB)	5400 cartridges	$50
Archive disk	320 GB	$300k per TB
Database disk	30 GB	$300k per TB

Parallel computer nodes	4 nodes	$75k
Database nodes	2 nodes	$25k
Parallel computer switch	1	$100k
Archive storage system	1	$150k per year
Database system	1	$100k per year

Table 3-6. Persistent archive for 1 billion digital objects

The total cost is $2.3 million. The largest single cost component is the tape drives for moving a total of 2 TB per day. If a larger container size is used, the duty cycle on the tape drives can be decreased, and the number of drives decreased proportionally. For a container size of 500 MB, the tape duty factor is about 3, the number of required drives is 7, and the system cost drops to $2 million. If containers are written sequentially to the same tape media, the drive duty factor can be reduced to less than a factor of 2, dropping the required number of tape drives to 4 and the cost to $1.9 million. The system is scalable, with a system twice the size costing $3.75 million. The parallel computer switch and software licenses costs do not change. A system one quarter the size costs $1.57 million.

An associated analysis is needed to quantify the number of nodes used to manage the data ingestion. For the E-mail collection described in section 5.2, the original data was provided in aggregated form to minimize the number of files that were ingested. The rate at which the meta-data was mined from the messages was about 70,000 messages per hour. On the order of 250,000 messages were loaded into the database per hour. If similar rates are sustained for the large scale persistent archive, accessioning 900,000 objects per day will take about 17 CPU hours, implying the need for a separate data ingestion node for the archive.

It is worth noting that CPU performance, disk and tape capacities, and disk and tape I/O rates will continue to improve. The expectation is that by year 2005, it will be possible to store a terabyte of data on a tape, and read the tape at 100 MB/sec. The types of storage media also may change dramatically, with disk platters replacing tapes, and eventually holographic storage replacing disk platters. The latter two advances will decrease the large latency associated with tapes, and make it possible to use as few as one quarter as many peripherals.

The metrics that have been discussed for a scalable persistent archive include cost, capacity, automation through use of tape robots, risk mitigation through use of backup procedures and multiple copies of critical data, and heterogeneity through use of separate systems for database and archival storage support. The corresponding tradeoffs are:

- **Cost versus scalability.** By eliminating the parallel computer switch, it is possible to build a cheaper system, but one which will be harder to expand as the total amount of data increases.

- **Cost versus complexity.** One can design a system in which data is migrated between two types of tape, with digital objects stored on very high capacity but higher latency tape, and the database meta-data stored on lower capacity but lower latency tape. The system is more complex, provides a cheaper solution as the size of the system is increased, but is not as cost effective for small systems.
- **Cost versus risk mitigation.** To safeguard data, two copies should be maintained on the cheapest archival storage media. This increases the cost of the system. The second copy can be migrated out of the tape robot and maintained on shelf, but the media costs are still incurred.
- **Access versus preservation.** The systems shown focus on the storage and update of data collections, and do not explicitly address data retrieval requirements. One approach for minimizing risk is to minimize the number of times a tape is handled. This is possible if frequently referenced data sets are maintained on disk. This increases the amount of disk space that is required.
- **Infrastructure independence versus proprietary format.** The update scenario assumes that an infrastructure dependent representation of the database tables is used. This should be augmented with the storage of an infrastructure independent representation of each collection on a periodic basis.
- **Automation versus operation cost.** The scenarios rely on the use of tape robots to minimize operational labor costs. Labor costs will increase if a second tape copy is maintained on shelf outside the robot, but will be partially balanced by decreasing the need for an additional tape robot. Note that the major labor cost will be incurred if robots are not used when the data is migrated to new media, requiring that all tapes be accessed by hand and read.
- **Automation versus risk mitigation.** Use of a tape robot provides a higher degree of assurance that tapes will not be dropped or damaged. However, note that even tape robots damage tapes occasionally. The experience at SDSC is roughly one damaged tape per 150,000 tape mounts.

3.2.2 Operational Systems

The principal components of a persistent archive are the archival storage system and the database management system. Both technologies are readily available from commercial vendors. However, no commercial products exist at the present time that provide a fully integrated system. Sites that require a persistent archive must decide how the integration can be best done for their usage requirements, and either implement the solution locally, outsource to a systems integrator, or find a vendor that provides a persistent archive service. All approaches incur some level of risk. Maintaining expertise sufficient to do the systems integration within an organization requires continual training of new personnel, and retention of the expertise while under competitive pressure from commercial institutions that need the same technology. Outsourcing assumes that the systems integrator will be available for the lifetime of the persistent archive. Relying on a commercial service requires assurance that the required level of persistence can be provided. When integrated database/archival storage systems are available commercially, it will become possible to run a persistent archive as an application, with a corresponding decrease in the level of expertise that is required.

Large scale archival storage systems and database systems are sufficiently complex that multiple staff are required both to maintain the systems and to migrate the systems to new technology. Large-scale production archives require support for the parallel computer operating system, support for the archival storage software, and testing support for new technology as it becomes available. Database systems require similar support for the parallel operating system and database administrators to manage the database software. For the persistent archive shown in Tables 3-5 and 3-6, the number of required staff is on the order of 10 full time systems analysts, plus two operators to monitor the system. Note that the archive at the San Diego Supercomputer Center is maintained by a staff of 5 system analysts. The operating labor costs can easily exceed the amortized hardware and software costs, assuming the technology is upgraded on a three to four year cycle.

The persistent archive scenarios have concentrated on the long-term storage of data collections. An equally important step is the ingestion process, for the creation of the collections. For the scale that is being discussed, the automation of the ingestion process becomes as important as the ability to migrate the data collections once they are archived. It is only feasible to consider collections of a billion objects if the attributes needed to identify each digital object can be generated by automated analysis of the input data stream. In sections 5 through 13, the ingestion process is described for each of the test collections, along with lessons learned on the ability to automate the process.

4. Collection support, general requirements

The process used to ingest a collection, transform it into an infrastructure independent form, and recreate the collection on new technology is shown schematically in Figure 4-1

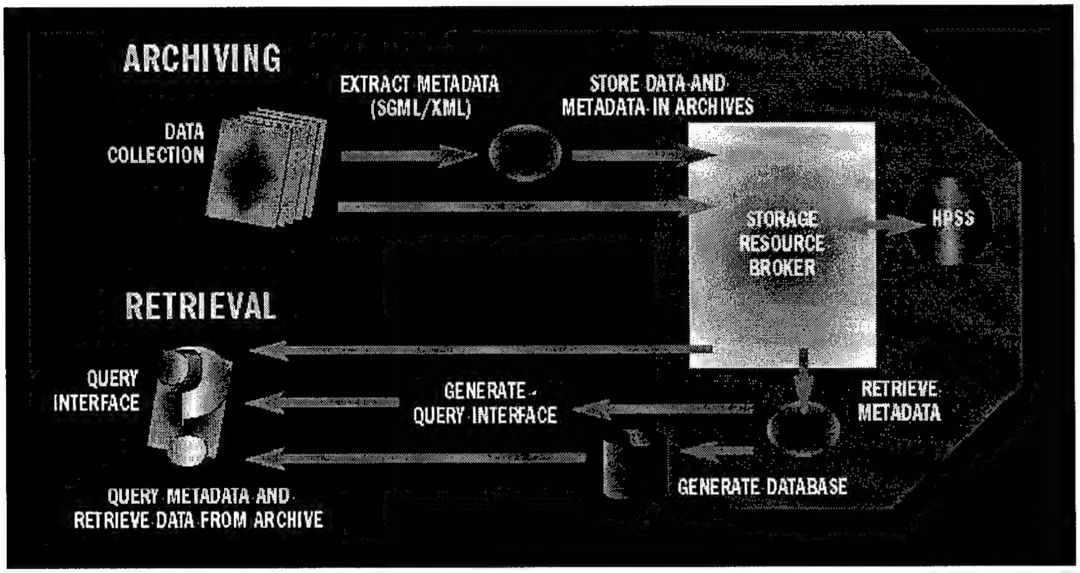


Figure 4-1. Persistent Collection Process

Two phases are emphasized, the archiving of the collection, and the retrieval or instantiation of the collection onto new technology. The diagram shows the multiple steps that are necessary to preserve digital objects through time. The steps form a cycle that can be used for migrating data collections onto new infrastructure as technology evolves. The technology changes can occur at the system-level where archive, file, compute and database software evolves, or at the information model level where formats, programming languages and practices change.

4.1 Collection Process Definition

The initial data set ingestion and collection creation can be seen as a process in which

(a) objects are captured, wrapped as XML digital objects, and categorized in a (relational) database system,
(b) the collection is ingested into an archival-storage system using containers to hold the digital objects along with all pertinent meta-data and software modules.

The migration cycle can be seen as the reverse of the ingestion process in which

(a) containers are brought out of deep-store and loaded into a (possibly NEW) database system (that can be relational or hierarchical or object oriented)
(b) the database is queried to form (possibly NEW) containers that are placed back into a (possibly NEW) archival storage system.

Note the similarities in steps (a) and (b) of the ingestion and migration processes. To facilitate migration one needs to make certain that the data formats and programs are also migrated forward. The key issue in the preservation process is that one needs to preserve the data collection in a manner that it can be recreated in a new form as and when required. Storage of objects in proprietary formats and/or machine/software-dependent formats will create problems in the future when the system and/or programs used to materialize these formats become obsolete.

In order to build a persistent collection, we consider a solution that "abstracts" all aspects of the data and its preservation. In this approach, data object and processes are codified by raising them above the machine/software dependent forms to an abstract format that can be used to recreate the object and the processes in any new desirable forms.

Data objects are abstracted by marking the contents of each digital object with tags that define the digital object structure. Tags are also used to mark the attributes that are used to organize the collection and define the collection context. An extensible mark-up language such as SGML or XML is used. XML (and SGML) has the advantage that the structure associated with the digital object can be defined through use of a Document Type Definition that can be applied across all objects within a collection. Processes are abstracted such that one can create a new "procedure" in a language of choice. Examples are the ingestion procedures themselves. They comprise 'abstract load modules' for building the collection. Similarly, the querying procedures can be represented as 'abstract mappings from a definition language to a query language' and visualization presentation procedures can be cast as 'style-sheet abstractions'.

The multiple migration steps can be broadly classified into a definition phase and a loading phase. The definition phase is infrastructure independent whereas the loading phase is geared towards materializing the processes needed for migrating the objects onto new technology.

We illustrate these steps by providing a detailed description of the actual process used to ingest and load a million-record data collection at SDSC. Note that the SDSC processes were written to use the available object-relational databases for organizing the meta-data. In the future, it may be possible to go directly to XML-based databases.

I. Ingestion/Creation Definition Phase

The SDSC infrastructure uses object-relational databases to organize information. This made data ingestion more complex by requiring the mapping of the DTD hierarchical representation onto a relational schema. Two aspects of the abstraction of objects need to be captured: relationships that exist in and among the data, and hierarchical structures that exist in the data. These were captured in two different types of abstractions: through a relational Data Definition Language (DDL), and through a hierarchical Document Type Definition (DTD). The relational abstraction facilitates querying about the meta-data, whereas the hierarchical abstraction facilitates presentation, storage and transportation. Hence, our process captured both of these aspects of digital objects. In the future, both aspects might merge due to emergence of XML-based database systems. In the model below, only the XML-DTD is stored as part of the abstract object; instead of storing the DDL, we store the procedure for creating a DDL from a DTD. A system-dependent DDL was created using the DTD and the DTD-to-DDL mapping procedure with the addition of system-specific information. The software that creates the system-dependent DDL comprises the instantiation program between the digital objects stored in the archive, and the collection that is being assembled on new technology.

Define Digital Object

- define meta-data
- define object structure (OBJ-DTD) --- (A)
- define OBJ-DTD to object DDL mapping --- (B)

Define Collection

- define meta-data
- define collection structure (COLL-DTD) --- (C)
- define COLL-DTD to collection DDL mapping --- (D)

Define Containers

- define packing format for encapsulating data and meta-data (examples are the AIP standard, Hierarchical Data Format, Document Type Definition)

II. Load Phase:

Create generator for the Database-DDL --- (E)

- [(A),(B),(C),(D),Target-system Info] ==> COLL-DDL --- (X)

Create generator for the database Loader --- (F)

- [(A),(C),(X),Target-system Info] ==> DB Load-module --- (Y)

Create generators for presentation interface and storage --- (G)

- [(A),(C),(X),Target-system Info] ==> SQL & Style-Sheet --- (Z)
- [(A),(C),(X),Target-system Info] ==> Archive Load-module --- (Z')

Generate Containers and Store

- Store also (A),(B),(C),(D),(E),(F),(G) as part of packed format.

In the ingestion phase, the relational and hierarchical organization of the meta-data is defined. No database is actually created, only the mapping between the relational organization and the object DTD. Note that the collection relational organization does not have to encompass all of the attributes that are associated with a digital object. Separate information models are used to describe the objects and the collections. It is possible to take the same set of digital objects and form a new collection with a new relational organization.

In the load phase, the mappings between the relational and hierarchical representations of the objects and collections are combined with the hierarchical information model to generate the relational representation for the new database on the target system. The information is encapsulated as a software script that can be used to create the new database tables for organizing the attributes.

A second script is created that is used to parse the digital objects that are retrieved from the archive, and load the associated meta-data into the new database tables.

Steps (A),(B),(C) and (D) can be interpreted as abstract mark-up formats for digital objects and steps (E),(F) and (G) can be interpreted as abstract procedures. The formats and procedures can be combined to support migration of the collection onto new software and hardware systems, as well as migration onto new information models or data formats and new procedure languages.

In a system-level migration process, a database is created using (X), the database is instantiated from the copy of the container(s) in the archival storage system using (Y), and the data in the database is stored into a new archival storage system using (Z').

In a format-level migration process, new versions of (A),(B),(C),(D) are created based on the prior values, a database is created using the original (X) and instantiated with the prior (Y), and then the data in the database is reformatted and stored in the archival storage system using the new (Z').
In a language-level migration process, new versions of (E),(F),(G) are created based on the original values, and stored as part of the packaged container. The data itself is not migrated.

4.2 Summary Information across all Collections

Table 4-1 indicates where each of the 9 collections are further documented in this report. It also matches each collection with one of the original Research and Development Plan and Schedule (RDPS) tasks, identifies the collection source, and specifies whether the collection was ingested into archival storage at SDSC and a demo developed.

Alias	Report	RDPS	Collection	Source	Ingestion	Demos
E-mail	Section 5 Appendix A	I. A.	E-Mail Postings	SDSC	Y	Y
Tiger92	Section 6 Appendix B	I. C.	Tiger/Line'92	Census	Y	
104th	Section 7 Appendix C	I. C.	104th Congress Bills	House	Y	
VAD97	Section 8 Appendix D	I. C.	105th Congress Roll Call Votes	House	Y	
EAP	Section 9 Appendix E	I. C.	Electronic Archive Project	NARA	Y	
Vietnam	Section 10 Appendix F	I. C.	Combat Area Casualties Current File -- CACCF	NARA	Y	Y
Patent	Section 11 Appendix	I. D.	Patent Data	USPTO	DOCT	DOCT Demo

	G					
AMICO	Section 12 Appendix H	I. C.	Image Collection (AMICO)	CDL California Digital Library	Y	Y
JTIC	Section 13 Appendix I	I. C.	Joint Interoperability Test Command	Defense	Y	Y

Table 4-1. Persistent Archive Demonstration Collections

For each of the demos developed, Table 4-2 further describes the type of features highlighted:

- *Preservation* deals with archival storage of the original collection. The labels are listed in bold for copying into HPSS -- **HPSS**, copying into HPSS through a database tablespace -- **UDB**, copying an XML-tagged version and associated DTD -- **XML**, and copying related relational database information -- **SQL**.
- *Access* specifies what technologies are used to query the archived collection. The labels are listed in bold for use of relational technology -- **SQL**, or XML-based technology -- **XMAS**.
- *Presentation* describes whether HTML stylesheets are used -- **HTML** or XML stylesheets are used -- **XSL**, or an **SQL** interface.
- *Consistency* gives an indication of the kind of quality assurance that was performed.

Alias	**Demos**			
	Preservation	**Access**	**Presentation**	**Consistency**
E-mail	HPSS / SQL / XML	SQL	HTML stylesheet	Checking for duplicate messages
Vietnam	HPSS / SQL / XML	SQL / XMAS	SQL command line presentation	Incomplete Info Checks Through SQL queries
AMICO	HPSS / SQL / XML	SQL / XMAS	XSL stylesheet HTML stylesheet	
JTIC	UDB / SQL	SQL	SQL command line presentation	

Table 4-2. Persistent Collection Demonstrations

Table 4-3 lists the *raw* size of each collection (by raw, we mean the size of the collection as assembled or delivered to us on CD-ROM, tape, or other), the number of records in the collection, the time to store the collection on the archive (we also indicate when additional time was spent TAR-ing the collection first), and the type of container technology used for storage purposes (digital objects are stored as ***files***, as ***records*** in containers, as ***database*** objects through use of UDB tablespaces, or as data handling objects within containers-- as done in the ***SRB*** technology).

	Raw Size	**# Records**	**Archival Time**	**Container Type**
E-mail	2.52 GB	1,000,000	1 h 02 m	Record / SRB

Tiger92	24.47 GB	50,951	**Tar**: 19h 28 m 18h 20m	Record
104th	0.32 GB	11,437	**Tar**: 0h 14m 0h 15m	file
VAD97	0.03 GB	1,288	**Tar**: 0h 03m 0h 2m 25s	file
EAP	0.84 GB	11,543	**Tar**: 0h 42m 0h 37m 13s	database
Vietnam	0.07 GB	58,181	**Tar**: 0h 03m 0h 2m 39s	database
Patent	150.00 GB	2,000,000	*Will be done in Aug.*	database
AMICO	0.12 GB	51	**Tar**: 0h 08m 0h 8m 06s	SRB
JTIC	0.38 GB	680	**Tar**: 0h 16m 0h 21m 29s	database

Table 4-3. Collection attributes

Finally, Table 4-4 characterizes each collection according to its suitability for XML and Relational representation, and captures in a few words the defining nature of the collection (***E-mail*** illustrates scalability and access, while ***AMICO*** illustrates images and complex meta-data, while ***Tiger92*** illustrates a topological spatial format suitable to long-term preservation but not conducive to interaction -- typically GIS vendors convert TIGER/Line'92 into a native interaction format.).

In Table 4-4, we distinguish meta-data and collection to characterize the suitability for relational representation..

- ***E-mail*** tags are suitable for a relational representation, however, the body of the E-mail message is not (Blob, attachment, unstructured text).
- ***EAP*** has relational meta-data but the digital object is an image.
- ***Patent*** is similar to *EAP*
- ***AMICO*** is similar to *Patent*
- ***JTIC*** does not seem particularly suited for any kind of relational representation.

	Suitability for XML Representation	**Suitability for Relational Representation**		**Main Feature**
		Meta-data	Object	
E-mail	Y	Y	*N*	Scalability & access
Tiger92	Y	Y	Y	spatial data storage vs. interaction
104th	Y	Y	Y	Formatted text
VAD97	Y	Y	Y	Fixed-record + HTML

EAP	Y	Y	*N*	Images & proprietary DB format
Vietnam	Y	Y	Y	Fixed-record structure
Patent	Y	Y	*N*	Compound documents
AMICO	Y	Y	*N*	Images & complex meta-data
JTIC	*N*	Y	*N*	Heterogeneous / binary data

Table 4-4. Information Models

Clearly, as shown in Table 4-4, *no single representation is good for everything*, whether meta-data-base, relational, or XML-based.

5. Collection support – E-mail postings

COLLECTION DESCRIPTION:

A collection of *1 million* records has been assembled. *E-mail* messages and *Newsgroup* (Usenet) messages were possible candidates. We settled on assembling the corpus from *Usenet groups*, collected at SDSC. The corpus of interest focuses on technical topics, including Computer Science, Science, Humanities, and Social Science. RFC 1036 provides a standard for Usenet messages that defines both required and optional attributes.

OBJECTIVES:

Demonstrate *preservation strategy* and *sustained access* to a 1 million record collection and look at the potential for upward *scalability.*

RELATED COLLECTIONS:

The main difference between *Newsgroup* records and *E-mail* records is that we are not concerned with attachments. To deal with attachments, lessons learned for the ***JTIC*** collection are relevant.

APPROACH TAKEN:

We created XML images of the 1 million records according to a well-defined E-mail DTD. A relational representation of the collection was also developed. The complete ingestion, archival storage, collection creation, and query access cycle was demonstrated. Sample SQL queries were shown against the resulting dynamically recreated collection.

IMPORTANT FINDINGS:

The1-million record E-mail collection was ingested, archived, and dynamically rebuilt within a single day. This was possible because all steps of the process were automated. The demonstration is scalable, such that archiving of 40-million E-mail records can be done within a month. The steps for the 1-million record demonstration included assembling the collection, tagging each message using XML, archival storage of the digital objects, instantiation as a new collection, indexing the collection, presentation through a Web interface, and support for queries against the collection. The required Usenet meta-data attributes form a core set that can be applied to all E-mail messages.

5.1 "Long-Term Preservation" Information Model

A typical example of a *raw* E-mail record is given in Figure 5-1. Note that unique tags are added to define the beginning and end of the record. This is required because it is possible for an E-mail message to contain an encapsulated E-mail message, making it difficult to create digital objects by explicit analysis of the complete collection.

```
____NARA_article_begin____:__________________
Path: news.sdsc.edu!newshub.csu.net!newshub.sdsu.edu!newsfeed.berkeley.edu!
  news.cis.ohiostate.edu!news.rootsweb.com!rootsweb-gw
From: Casivers@aol.com
Newsgroups: soc.genealogy.hispanic
Subject: Passenger Lists for Ships from Spain To Cuba
Date: 22 Mar 1999 16:20:37 -0800
Organization: RootsWeb Genealogical Data Cooperative
Lines: 7
Message-ID: <2376321.36f6de03@aol.com
NNTP-Posting-Host: localhost
Mime-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
X-Trace: bl-1.rootsweb.com 922148437 3147 127.0.0.1 (23 Mar 1999 00:20:37 GMT)
X-Complaints-To: usenet@news.rootsweb.com
NNTP-Posting-Date: 23 Mar 1999 00:20:37 GMT
Xref: news.sdsc.edu soc.genealogy.hispanic:3156

Does anyone know where I can get passengers lists for
ships that transported Spaniards to Cuba circa 1860's?
Any help would be appreciated.
Thanks,
Cheryl Sanchez-Sivers
____NARA_article_end____:____________________
```

Figure 5-1. Example of raw E-mail message

Object-Level Structure: XML DTD.

A DTD was derived which reflects the *RFC1036* structure described in Appendix A. Please note that each of the *required*, *optional*, and *other* keyword items is associated with a *seqno* attribute. This attribute is used to record information on the sequence in which the various keywords appear in the original document, since the order of appearance of keywords may be different in different documents. The DTD for the E-mail messages is given in Figure 5-2.

```
<!ELEMENT rfc1036_mesg (headers, body)>

<!ELEMENT headers (required_headers, optional_headers, other_headers)>
```

```
<!ELEMENT body #PCDATA>

<!ELEMENT required_headers (From, Date, Newsgroups, Subject, Message-ID, Path)>
<!ELEMENT optional_headers (Folloup-To?, Expires?, Reply-To?, Sender?, References?,
                            Control?, Distribution?, Keywords?, Summary?, Approved?,
                            Lines?, Xref?, Organization?)>
<!ELEMENT other_headers other+>

<!-- 6 required header keywords -->
<!ELEMENT From              #PCDATA>
<!ELEMENT Date              #PCDATA>
<!ELEMENT Newsgroups         #PCDATA>
<!ELEMENT Subject           #PCDATA>
<!ELEMENT Message-ID        #PCDATA>
<!ELEMENT Path              #PCDATA>

<!ATTLIST From         seqno   CDATA #REQUIRED>
<!ATTLIST Date         seqno   CDATA #REQUIRED>
<!ATTLIST Newsgroups   seqno   CDATA #REQUIRED>
<!ATTLIST Subject      seqno   CDATA #REQUIRED>
<!ATTLIST Message-ID   seqno   CDATA #REQUIRED>
<!ATTLIST Path         seqno   CDATA #REQUIRED>

<!-- 13 optional header keywords -->
<!ELEMENT Followup-To       #PCDATA>
<!ELEMENT Expires           #PCDATA>
<!ELEMENT Reply-To          #PCDATA>
<!ELEMENT Sender            #PCDATA>
<!ELEMENT References        #PCDATA>
<!ELEMENT Control           #PCDATA>
<!ELEMENT Distribution      #PCDATA>
<!ELEMENT Keywords          #PCDATA>
<!ELEMENT Summary           #PCDATA>
<!ELEMENT Approved          #PCDATA>
<!ELEMENT Lines             #PCDATA>
<!ELEMENT Xref              #PCDATA>
<!ELEMENT Organization      #PCDATA>

<!ATTLIST Followup-To       seqno   CDATA #REQUIRED>
<!ATTLIST Expires           seqno   CDATA #REQUIRED>
<!ATTLIST Reply-To          seqno   CDATA #REQUIRED>
<!ATTLIST Sender            seqno   CDATA #REQUIRED>
<!ATTLIST References        seqno   CDATA #REQUIRED>
<!ATTLIST Control           seqno   CDATA #REQUIRED>
<!ATTLIST Distribution      seqno   CDATA #REQUIRED>
<!ATTLIST Keywords          seqno   CDATA #REQUIRED>
<!ATTLIST Summary           seqno   CDATA #REQUIRED>
<!ATTLIST Approved          seqno   CDATA #REQUIRED>
<!ATTLIST Lines             seqno   CDATA #REQUIRED>
<!ATTLIST Xref              seqno   CDATA #REQUIRED>
<!ATTLIST Organization      seqno   CDATA #REQUIRED>

<!-- other header keywords -->
<!ELEMENT other        #PCDATA>

<!ATTLIST other
```

```
        keyword         CDATA #REQUIRED
        seqno           CDATA #REQUIRED>
```

Figure 5-2. DTD for E-mail message.

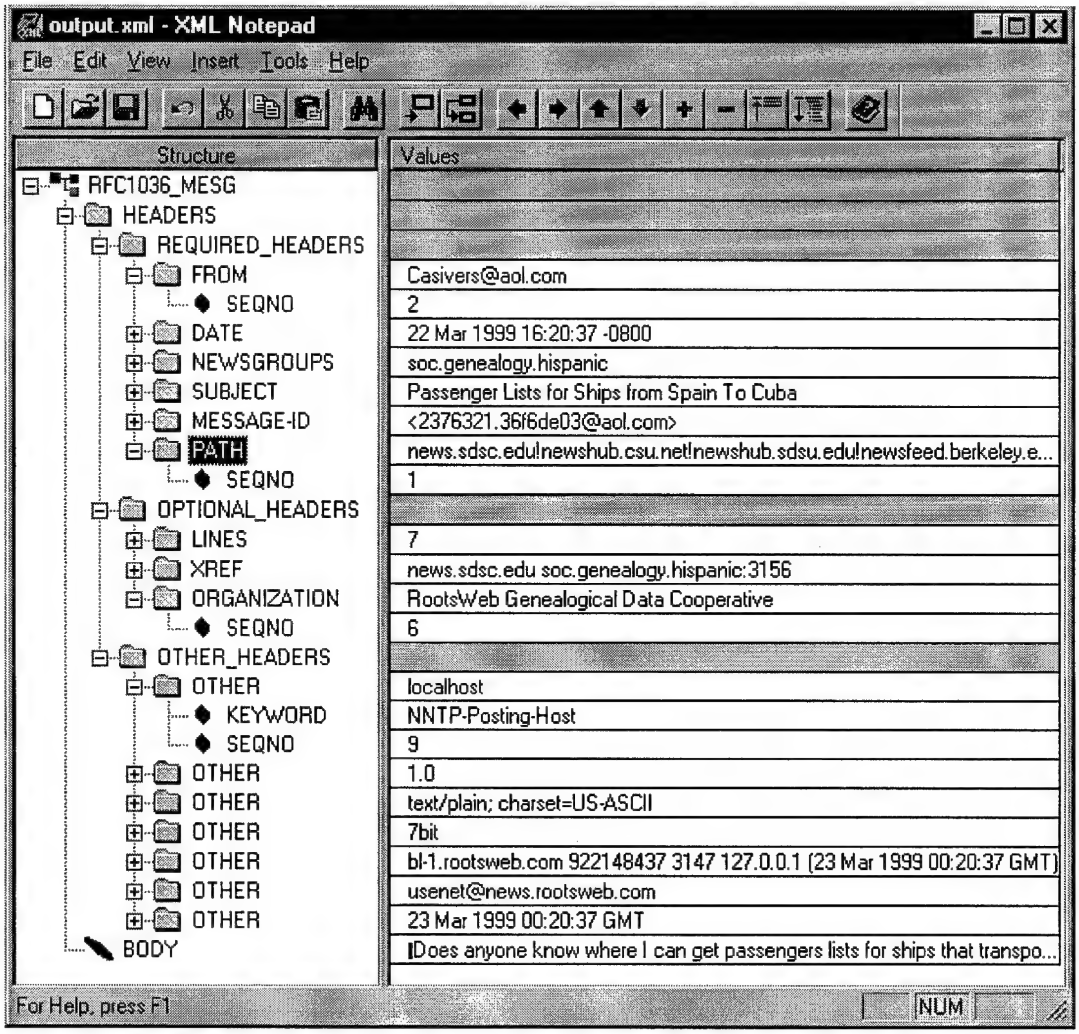
The Newsgroup message shown previously in Figure 5-1 can be displayed using Microsoft Notepad, an XML viewer, by applying the DTD shown in Figure 5-2. This provides the ability to impose a presentation style on the objects in a collection. The result is shown in Figure 5-3.

Object-Level Structure: Relational Schema.

Loading sources with regular structure into a relational database (RDB) has several benefits:

- Inconsistencies in the data can be automatically detected using the RDB's built-in *consistency checks* (data types, uniqueness of keys, referential integrity, etc.)
- Powerful ad-hoc SQL queries can be used to further *clean* the data from inconsistencies
- Interesting information from the collection can be *mined*
- Different versions of the collection can be *compared*
- Using an RDB-to-XML wrapper provides an XML view on the collection.

This same approach is used for the CACCF (Vietnam War Casualties collection). In the case of the Newsgroup records, however, the underlying Relational Database structure is much more complex as not all attributes are required. The semi-structured nature of the E-mail messages is more easily represented with an XML hierarchical representation.



Structure	Values
RFC1036_MESG	
HEADERS	
REQUIRED_HEADERS	
FROM	Casivers@aol.com
SEQNO	2
DATE	22 Mar 1999 16:20:37 -0800
NEWSGROUPS	soc.genealogy.hispanic
SUBJECT	Passenger Lists for Ships from Spain To Cuba
MESSAGE-ID	<2376321.36f6de03@aol.com>
PATH	news.sdsc.edu!newshub.csu.net!newshub.sdsu.edu!newsfeed.berkeley.e...
SEQNO	1
OPTIONAL_HEADERS	
LINES	7
XREF	news.sdsc.edu soc.genealogy.hispanic:3156
ORGANIZATION	RootsWeb Genealogical Data Cooperative
SEQNO	6
OTHER_HEADERS	
OTHER	localhost
KEYWORD	NNTP-Posting-Host
SEQNO	9
OTHER	1.0
OTHER	text/plain; charset=US-ASCII
OTHER	7bit
OTHER	bl-1.rootsweb.com 922148437 3147 127.0.0.1 (23 Mar 1999 00:20:37 GMT)
OTHER	usenet@news.rootsweb.com
OTHER	23 Mar 1999 00:20:37 GMT
BODY	Does anyone know where I can get passengers lists for ships that transpo...

Figure 5-3. Formatted message using XML DTD

We define 3 *hand-crafted* tables for the Newsgroup collection for the relational database:

- The first table contains all the *required* and *optional* header field information supported in RFC1036. Additionally there is an *internalMsgId* used for cross-references with other tables. *sequence* numbers are used to show the sequence in which the fields appeared in the original message.
- The second table contains facilities for storing *other* header fields not supported by rfc1036.
- The third table contains 'systemic' information about how the body of messages are stored. The *dataid* field refers to a "file" or "container" id. The *posInContainer* field is the offset of the start of that record's body text. The *sizeOfMsg* field is the length in bytes of that record's body text.

What we ended up doing in practice for storing the collection, was to concatenate together subsets of the text from the bodies of the Newsgroup messages, not into one huge file over the million record collection, but into smaller files that would hold the bodies of 40,000 records at a time. Hence we created 25 files or "containers".

In the current implementation of the data handling system, a “container” file can be registered with SRB/MCAT (the SDSC data handling system). SRB/MCAT provides access mechanisms for retrieving individual messages from such containers.

For loading the Newsgroup records into a relational database (in this case: Oracle), the following schema was used:

```
create table ngrps_headers_core (
   internalMsgId       integer        not null,
   FromInfo            varchar(200)   not null,
   MsgDate             varchar(50)    not null,
   Newsgroups          varchar(1900)  not null,
   SubjectInfo         varchar(1900)  not null,
   MessageId           varchar(200)   not null,
   PathInfo            varchar(1900)  not null,
   FollowupTo          varchar(1900),
   ExpiresOn           varchar(50),
   ReplyTo             varchar(200),
   SenderInfo          varchar(200),
   ReferencesInfo      varchar(1900),
   ControlInfo         varchar(1900),
   DistributionInfo    varchar(500),
   KeywordsInfo        varchar(1900),
   SummaryInfo         varchar(1900),
   ApprovedInfo        varchar(500),
   LinesOfEmail        integer,
   XrefInfo            varchar(500),
   OrganizationInfo    varchar(500),
   FromSeqNum          integer,
   MsgDateSeqNum       integer,
   NewsgroupsSeqNum    integer,
   SubjectSeqNum       integer,
   MessageIdSeqNum     integer,
   PathSeqNum          integer,
   FollowupToSeqNum    integer,
   ExpiresSeqNum       integer,
   ReplyToSeqNum       integer,
   SenderSeqNum        integer,
   ReferencesSeqNum    integer,
   ControlSeqNum       integer,
   DistributionSeqNum  integer,
   KeywordsSeqNum      integer,
   SummarySeqNum       integer,
   ApprovedSeqNum      integer,
   LinesSeqNum         integer,
   XrefSeqNum          integer,
   OrganizationSeqNum  integer,
   primary key (internalMsgId),
   unique (MessageId,MsgDate)
);

create table ngrps_headers_othr (
   internalMsgId       integer          not null,
   HdrKeyName          varchar(50),
   HdrKeyValue         varchar(2000),
   HdrKeySeqNum        integer,
          foreign key (internalMsgId) references ngrps_headers_core
      (internalMsgId)
);

create table container_info (
    internalId          integer         not null,
    data_id             integer         not null,
    posInContainer      integer,
    sizeOfMsg           integer,
    foreign key (internalId) references ngrps_headers_core(internalMsgId),
    foreign key (data_id) references MDAS_AD_REPL(data_id)
);
```

Figure 5.4. Relational Schema for storing E-mail collection

Note that the only fields that are required are the six Usenet required attributes, the internal message ID that uniquely names each message, and the data ID that defines the container in which the message is stored.

Collection-Level Structure

The structure at the *collection level* can be described by the following XML DTD:

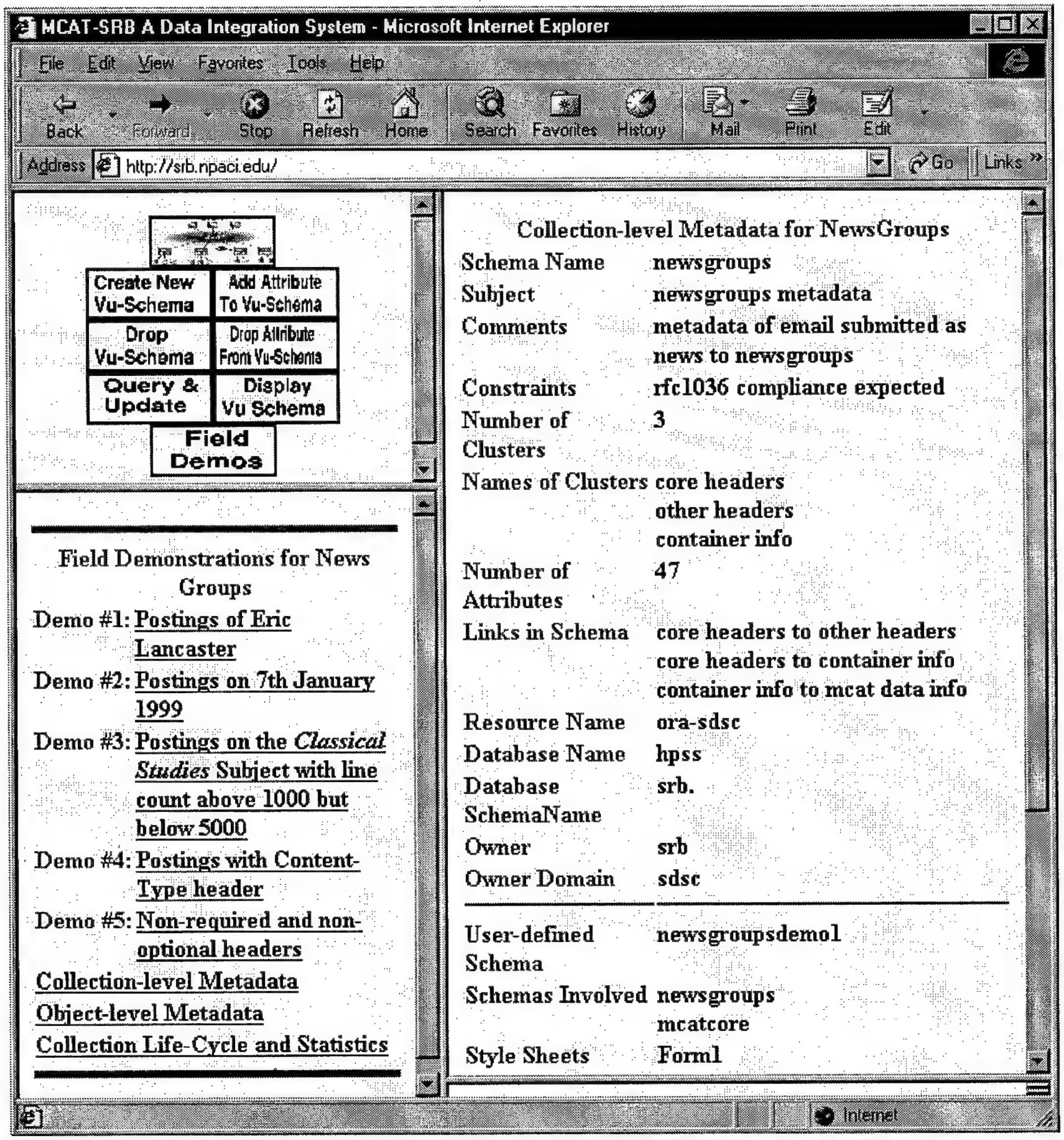
```
<!ELEMENT newsgroup_collection (rfc1036_mesg)*>        list of newsgroup messages
```

Figure 5.5. Collection level definition of the E-mail collections

This is of course a trivial case where we simply create a list of individual Newsgroup records. Additional "meta-data" or ELEMENT and ATTRIBUTE information could be stored here if available. Examples would be the source of data for the collection, the collection ingestion date, and the provider of the collection.

For the database representation of the collection, we define a set of generic meta-data attributes.

The following screen snapshot shows a subset of collection-level meta-data about the ***E-mail*** collection. Typical attributes are: schema name, subject, comments, constraints, number of SRB clusters, number of attributes, type of attributes, etc.


MCAT-SRB A Data Integration System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://srb.npaci.edu/
Go Links »
Create New Vu-Schema
Add Attribute To Vu-Schema
Drop Vu-Schema
Drop Attribute From Vu-Schema
Query & Update
Display Vu Schema
Field Demos
Internet

Field Demonstrations for News Groups

Demo #1: Postings of Eric Lancaster

Demo #2: Postings on 7th January 1999

Demo #3: Postings on the *Classical Studies* Subject with line count above 1000 but below 5000

Demo #4: Postings with Content-Type header

Demo #5: Non-required and non-optional headers

Collection-level Metadata

Object-level Metadata

Collection Life-Cycle and Statistics

Collection-level Metadata for NewsGroups

Schema Name	newsgroups
Subject	newsgroups metadata
Comments	metadata of email submitted as news to newsgroups
Constraints	rfc1036 compliance expected
Number of Clusters	3
Names of Clusters	core headers other headers container info
Number of Attributes	47
Links in Schema	core headers to other headers core headers to container info container info to mcat data info
Resource Name	ora-sdsc
Database Name	hpss
Database SchemaName	srb.
Owner	srb
Owner Domain	sdsc
User-defined Schema	newsgroupsdemo1
Schemas Involved	newsgroups mcatcore
Style Sheets	Form1

Figure 5.6. Schema level attributes used to define the collection

5.2 Ingestion process

We have automated the ingestion process from a live feed by developing a harvesting Perl script: ***harvest.pl***. This script when first executed, goes to the location where "live" E-mail messages are being stored on disk after they are received by SDSC's Newsgroup Server (/misc/news/spool on the Server machine in question). The script then recursively descends through each newsgroup subdirectory (in our case we limited the harvesting to four subdirectories: comp, humanities, sci, and soc), and retrieves posted messages (stored as individual files) concatenating them into a large buffer, *collection.raw*, that will later be archived to HPSS.

An enhancement we made, is that *harvest.pl*, iterates through this process until it has collected the 1,000,000 desired records. This was accomplished over a period of 3 1/2 weeks. Because messages typically get purged from the News Server after 3 days, the *harvest.pl* is awakened every 2 days (*cron* job or timeout can be used) and incrementally adds fresh messages to the running buffer. To accomplish this task, an internal database is built that logs which newsgroup names it visits and what the current *messages-posted* range is, so that it can in later runs only grab the differential or new messages that have arrived.

Note that while harvesting a "raw" aggregate collection file from 1 million individually harvested E-mail messages, unique delimiters were added to mark the start and end of records. The earlier E-mail example proposes two such delimiters. Example:

```
____NARA_article_begin____:__________________
____NARA_article_end____:____________________
```

A separate script, ***raw2xml.pl***, converts the delimited E-mail records into XML records and transforms *collection.raw* into *collection.xml*. Note that when creating XML digital objects, filtering was carried out on "<", "@", and ">" characters (transformed into "&lt;"&#64", "&gt"), , which have special meaning in XML. Also the text body of each message was encapsulated in one or more series of XML "<![CDATA[" and "]]> header and footer sequences.

Finally, a third Perl script, ***xml2sql.pl***, is used for dynamic instantiation of the archived XML collection, for presentation purposes using a relational model approach. Note that while loading records into the database, it was observed that roughly 17% of all messages were duplicates, corresponding to cross-listings. We chose to accommodate duplicates as multiple records in our demo.

Ingestion Statistics (real times @chagall.sdsc.edu).

The ingestion process was carried out on a workstation (chagall.sdsc.edu). The system was an SGI Indigo 2 (MIPS R10000 Processor, 195, Memory size: 128 Megabytes).

collection size:	2.52 GB
#files:	1 Million
#records/file:	6 required, 13 optional, variable other
Time to assemble collection: **SDSC Newsgroup --> Collection.raw**	12h (1 h 53m to collect 156,000 messages)
Time to convert: **Collection.raw --> Collection.xml**	1h 39m
Time to archive: **Collection.raw --> HPSS**	1h 02m
Time to archive: **Collection.xml --> HPSS**	1h 29m

Time to unarchive: **Collection.xml+DDL --> Disk**	* variable -- depends on load & caching policies on HPSS
Time to convert: **Collection.xml+DDL --> collection.sqlload**	2h 40m
Time to load: **Collection.sqlload --> DBMS**	4h
Time to optimize queries: **Index creation on tables**	4h
Time to query: **SQL query to Web interface**	A few seconds for typical queries
TOTAL LIFECYLE TIME	**< 27h (Just over a DAY!)**

Figure 5.7. Ingestion Times for the E-mail Collection

Assembly and conversion scripts were written in *Perl* and run on Chagall, a fairly slow single-processor SGI Indigo2.

The **complete lifecycle** for: Assembling, Tagging, Archiving, Instantiating, and Querying of 1 million Newsgroup records, takes roughly **27 hours.** The steps for assembling, tagging, and archiving took only 16 hours.

5.3 Storage requirements

The raw collection *collection.raw* of 1 million messages is *2.52 GB* in size. The tagged version, 1 million XML records, *collection.xml*, is *3.45 GB*. This represents an expansion factor of *37%*, which can be reduced using appropriate (shorter) tag names.

Because we chose to make *25* message body data files (1 for every 40,000 messages), the average size of each file was: *64.3 MB*. Each of these 25 files is stored as a SRB container file.

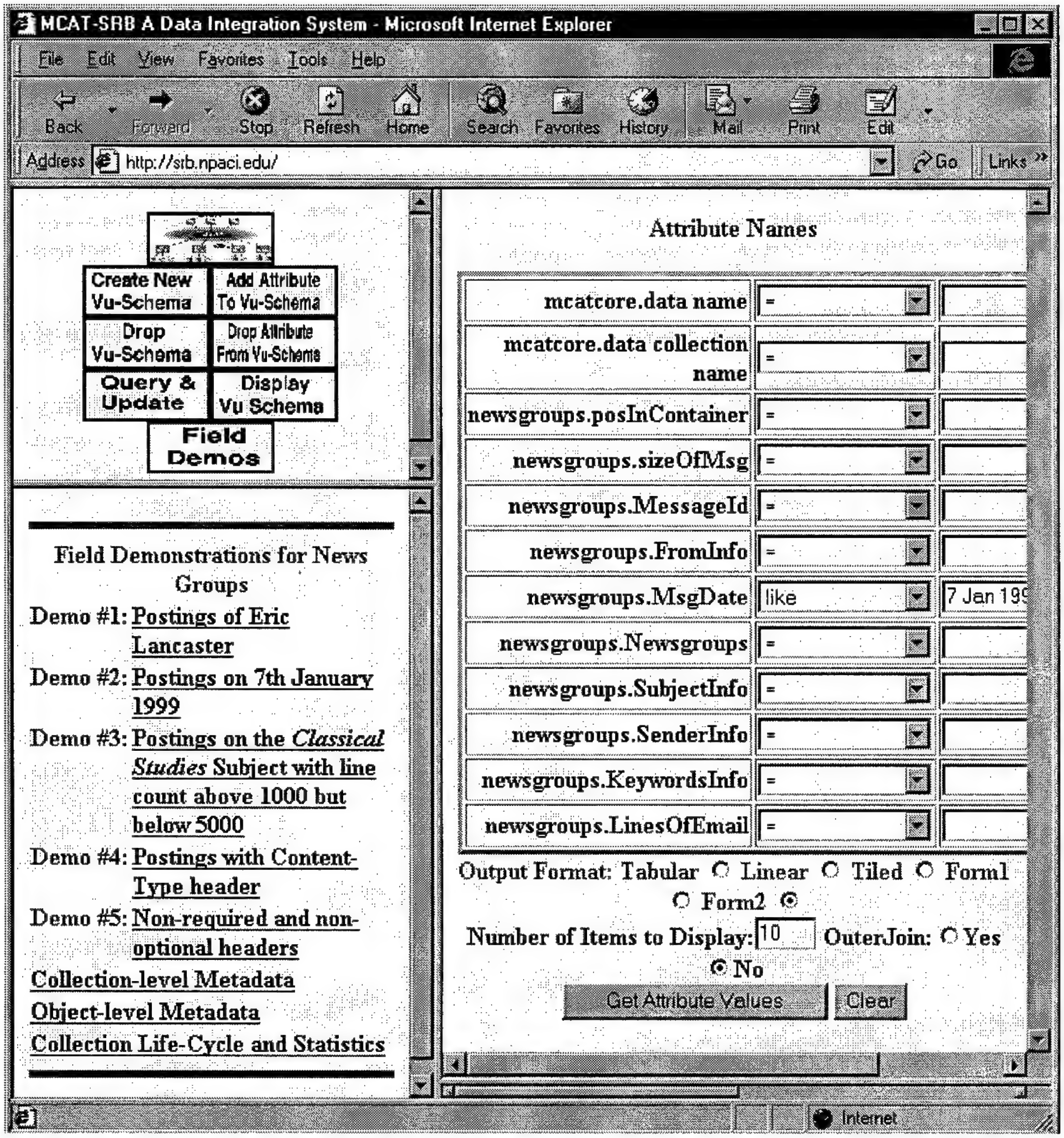
5.4 Data access requirements

The need for faster access and indexing to the database image is necessary when recreating the data collection. The 3 database tables were indexed with statements such as:

```
create index nhcFrom on ngrps_headers_core (FromInfo) unrecoverable;
...
```

Figure 5.8. Indexing commands

As expected index creation brought querying times down back to a few seconds (as opposed to the initial 20 minutes or so). A demonstration of the archive process has been created. The main presentation demo has built-in queries, such as "Show all Postings on 7th January, 1999) to demonstrate the ability to retrieve arbitrary messages from the recreated message collection. One can also specify particular attribute values to pull out a single message. In Figure 5.9, the Web-based user interface is presented which supports attribute based access to the collection.


MCAT-SRB A Data Integration System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://srb.npaci.edu/
Go Links
Create New Vu-Schema
Add Attribute To Vu-Schema
Drop Vu-Schema
Drop Attribute From Vu-Schema
Query & Update
Display Vu Schema
Field Demos
Get Attribute Values
Clear
Internet

Field Demonstrations for News Groups

Demo #1: Postings of Eric Lancaster

Demo #2: Postings on 7th January 1999

Demo #3: Postings on the *Classical Studies* Subject with line count above 1000 but below 5000

Demo #4: Postings with Content-Type header

Demo #5: Non-required and non-optional headers

Collection-level Metadata

Object-level Metadata

Collection Life-Cycle and Statistics

Attribute Names

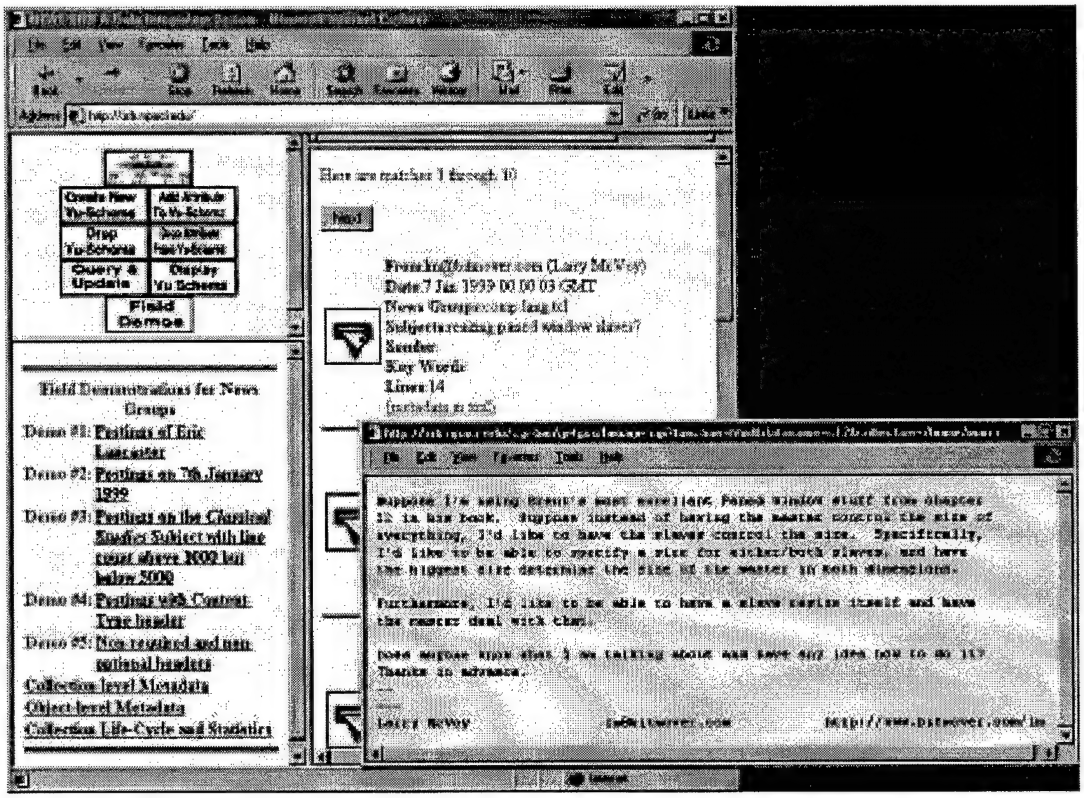
mcatcore.data name	=	
mcatcore.data collection name	=	
newsgroups.posInContainer	=	
newsgroups.sizeOfMsg	=	
newsgroups.MessageId	=	
newsgroups.FromInfo	=	
newsgroups.MsgDate	like	7 Jan 19
newsgroups.Newsgroups	=	
newsgroups.SubjectInfo	=	
newsgroups.SenderInfo	=	
newsgroups.KeywordsInfo	=	
newsgroups.LinesOfEmail	=	

Output Format: Tabular ○ Linear ○ Tiled ○ Form1 ○ Form2 ●

Number of Items to Display: 10 OuterJoin: ○ Yes ● No

Figure 5.9. Web-based interface for accessing the E-mail collection

In Figure 5.10, the results of the above query are presented. The screen snapshot shows how results are posted in groups of 10 according to a default HTML style sheet. Notice that when clicking on the mailbox icon, the record's text body shows up in a separate blue-tinted window. Also, if one were to click on the *(meta-data in XML)* link, a separate XML browser would be invoked, allowing the display of the XML fields for that record. This last feature is interesting as it shows how one could use database filtering mechanisms to transform relational data back into tagged XML data on-the-fly.



Here are matches 1 through 10

From:lm@bitmover.com (Larry McVoy)
Date:7 Jan 1999 00:00:03 GMT
News Groups:comp.lang.tcl
Subject:resizing paned window slaves?
Sender:
Key Words:
Lines:14
(metadata in xml)

Field Demonstrations for News Groups

Demo #1: Postings of Eric Lancaster
Demo #2: Postings on 7th January 1999
Demo #3: Postings on the Classical Studies Subject with line count above 3000 but below 5000
Demo #4: Postings with Content-Type header
Demo #5: Non-required and non-optional headers
Collection level Metadata
Object-level Metadata
Collection Life-Cycle and Statistics

```
Suppose I'm using Brent's most excellent Paned Window stuff from chapter
32 in his book.  Suppose instead of having the master control the size of
everything, I'd like to have the slaves control the size.  Specifically,
I'd like to be able to specify a size for either/both slaves, and have
the biggest size determine the size of the master in both dimensions.

Furthermore, I'd like to be able to have a slave resize itself and have
the master deal with that.

Does anyone know what I am talking about and have any idea how to do it?
Thanks in advance.
--
---
Larry McVoy              lm@bitmover.com           http://www.bitmover.com/lm
```

Figure 5.10. Results of E-mail collection query

5.5 Long term preservation requirements

Even though Newsgroup records do not contain attachments, there are ways to generate associated binary data that can be archived. For example,
(test-xmtp@jabr.ne.mediaone.net)
is a site to which one can send binary data in MIME format and it will send back an XML document where binary data is base64 encoded (uuencode). This gives a flavor of the kinds of encoding schemes one could consider when dealing with attachments.

Our initial goal was to show scalability beyond 40 million records. Based on the ability to ingest and archive the 1-million record collection in 16 hours, it should be possible to accession 40 million records in a month, using a single workstation.

6. Collection support – TIGER/Line '92 (Census Bureau)

COLLECTION DESCRIPTION:

The TIGER/Line Files are extracts of selected geographic and cartographic information from the Census Bureau's TIGER (Topologically Integrated Geographic Encoding and Referencing) System. They represent geographic features, such as roads, railroads, rivers, lakes, political boundaries, and census statistical boundaries. for all of the United States. TIGER/Line '92, while superseded by TIGER/Line 1997 and 1998, is a very important collection as it provides a link between the 1980 and 1990 Census Geography.

OBJECTIVES:

Demonstrate preservation of spatial data with a collection whose records are suitable for long-term retention but not necessarily for interactive manipulation. Note that ***Tiger92*** files are not "graphic images of maps". To make use of them, a user must have mapping or Geographic Information System (GIS) software that can import TIGER/Line data.

RELATED COLLECTIONS:

Because many of the records conform to well-defined fixed-length data dictionaries, this collection would relate to some aspects of the ***Vietnam*** collection.

APPROACH TAKEN:

The core ***Tiger92*** files and associated documentation can be organized using XML. A corresponding well-defined DTD is sketched.

IMPORTANT FINDINGS:

A DTD is proposed which may be well suited for describing data collections used with mapping software. This DTD "links" together XML representations of:

- *4 nation-wide lookup tables* (State and City FIPS codes, State abbreviations, Census Feature Classes, Urbanized Area Codes)
- *58 State-based documents* (state + statistically equivalent entity such as the District of Columbia and Puerto Rico), where each state document *contains 2 statewide lookup tables* (Census Entities and School district), and up to *14 county files* per county, each with a well defined format.

6.1 Information model

Analysis of the ***Tiger92*** collection format as detailed in *Appendix B*, leads us to the following observations:

Challenges:

- The TIGER/Line'92 data is a series of excerpts from an actual database ("LIVE TIGER"). Thus using this "flattened" version to create an image useful for long-term retention is an added complication. *An alternative*, if feasible, would be to work directly with the Census Bureau to capture a snapshot of the Live Tiger database for archival storage.

- The overall collection data is heterogeneous (delivered across a set of CDs). It contains:
  - *Text files*
  - *Word Processor files*: WordPerfect, Macintosh formats, Word
  - *Database files*: .DBF
  - *Graphics files*
  - *Binary executables*: *LandView* software program, a standalone TIGER/Line viewing program (not used by GIS practitioners though)
  - Up to14 record files / county, following the previous data dictionary format
- Since each CD-ROM is autonomous, many common files are duplicated
- The underlying topological spatial model used is not the format used for interaction with the data: typically each GIS vendor implements its own translators
- The associated documentation contains much of the semantics not available in the 14 county files
- The documentation may have key information in the appendices that is not available in the data directory portion of the CD-ROMs (e.g. FIPS code table, standard abbreviations, census feature class codes (CFCC), urbanized area codes and names).

Potential:

- When focusing on the core elements of the CD-ROM data: *county topological information* & *documentation* & *lookup tables*, the use of XML appears to be a viable alternative for a self-documenting long-term retention solution.
- Because of the underlying topological model, each series of county files is self-contained, in the sense that they can be independently assembled with neighboring county files.

Proposed Solutions:

- Explore the use of *DDI* (see below) for conversion of the documentation to XML DTDs
- Use *XML* DTDs for all topological county files
- Use *XML* DTDs for all lookup/code tables
- Create a *DTD* that will integrate all of these XML-tagged subcomponents: the implicit relationship between fields as loosely described in the documentation will be explicitly created for long-term preservation purposes
- Explore the use of emerging *XML linking* standards (*XPointer* and *Xlink* -- http://www.w3.org/XML/) for establishing explicit relationships between topological elements and documentation chunks, as well as "foreign keys" or common attributes across the 14 county files (relationships that are only described in the documentation).

XML Documentation

An emerging XML documentation standard can be used to simplify the ingestion of the documentation text: DDI (http://www.icpsr.umich.edu/DDI/). The Data Documentation Initiative (DDI) is a project designed to develop an XML Document Type Definition (DTD) for data documentation. The DDI standard is meant to be a new meta-data standard for social science documentation. It can serve as a structured codebook standard that can be used as an interchange format. DDI is being considered for the ***TIGER/Line'92 Technical Manual***.

DTD/XML for Tiger92:

Heterogeneity of the data on CD-ROM notwithstanding, it appears that when one factors out the *LandView* related files (binaries, documentation, .DBF files, etc.), the core files that remain are all ASCII-based and can be readily documented and preserved using an integrated DTD (*LandView* is the viewing software that is provided).

For each state (58 corresponding state FIPS codes), there are one or more counties with up to 14 county records (note that for 1992 there are a total of 3,428 counties):

- **CTY**: Each of the 14 county files has a fixed length record format with well-defined fields:

Record Number	**Record Length** (number of characters)	**Number of Fields**
1:	247	50
2:	208	24
3:	126	33
4:	63	9
5:	52	8
6:	85	14
7:	74	11
8:	36	8
A:	108	25
F:	73	20
G:	52	13
I:	52	11
P:	44	8
R:	46	9

Figure 6.1. County file format

For each state, there is an associated *TGR92Sst.NAM* and *SCHOLDst.NAM* file:

- **Census_Entities**: *TGR92Sst.NAM* is a geographic reference file, which contains names & codes for the census geographic entities:
  - Minor civil divisions
  - Places
  - Urbanized areas
- **School_District**: *SCHOLDst.NAM* contains school district codes & names

Each of these files has a data dictionary (see Appendix B). The first file is in ASCII text, formatted in fixed length *80-character records*, the second file type has a record length of *39 characters*. Each file also has an associated documentation file (.DOC) that could be the basis of an associated DDI XML documentation file, if useful.

All states make use of 4 global code tables:

1. **Codes_StCty**: FIPS State & County Codes (Technical Manual, Appendix A)

ST	COUNTY	AREA NAME	ADR#
06	073	San Diego County	Full (85% to 100%)

2. **Codes_StdAbbrevs**: Standard Abbreviations (Technical Manual, Appendix D)

Feature Type	STD Abbrev.	Short Abbrev.	USPS Ref.	Spanish	Translation
Calle	Calle	C	CLL	S	Street

3. **Codes_CensusFeatureClass**: Census Feature Class Codes (Technical Manual, Appendix E)

Census Feature Class Code	Text
D29	Shelter or Mission

4. **Codes_UrbanizedArea**: Urbanized Area Codes & Names (Technical Manual, Appendix G)

Urbanized Area Code	Name
7320	San Diego

Object Level Structure: Sketch of an XML DTD.

XML elements are defined for all of the above categories. Specific element attributes would, in our first pass sketch, mirror the corresponding data dictionary attributes, and would closely resemble the "object level structure XML/DTD for the CACCF collection, as all of the attributes appear to be REQUIRED. Hence, repeated individual data dictionary items are omitted in the DTD sketched below.

```
<!-- 14 County TIGER/Line'92 record elements -->
<!ELEMENT CTY_1>
<!ELEMENT CTY_2>
<!ELEMENT CTY_3>
<!ELEMENT CTY_4>
<!ELEMENT CTY_5>
<!ELEMENT CTY_6>
<!ELEMENT CTY_7>
<!ELEMENT CTY_8>
<!ELEMENT CTY_a>
<!ELEMENT CTY_f>
<!ELEMENT CTY_g>
<!ELEMENT CTY_i>
<!ELEMENT CTY_p>
<!ELEMENT CTY_r>

<!-- TGR92Sst.NAM and SCHOLDstNAM record -->
<!ELEMENT Census_Entities>
<!ELEMENT School_District>

<!-- Global Code Table Entities -->
<!ELEMENT Codes_StCty>
<!ELEMENT Codes_StdAbbrevs>
<!ELEMENT Codes_CensusFeatureClass>
<!ELEMENT Codes_UrbanizedArea>

<!-- ATTLIST's for each of the above elements... -->
<!ATTLIST ...  NOT FURTHER DETAILED IN THIS SKETCH ...>
```

Figure 6.2. DTD structure for Tiger/Line data objects

Collection Level Structure.

The structure at the *collection level* can be described by an XML DTD, shown in Figure 6.3, which links XML elements defined in the previous section:

```
<!ELEMENT TIGERLine92  (codeTables, states)>         list of global tables and States

<!ELEMENT codeTables (                               global code tables
          Codes_StCty,
          Codes_StdAbbrevs,
          Codes_CensusFeatureClass
          Codes_UrbanizedArea)>
```

```
<!ELEMENT states (state+)>
<!ELEMENT state (                                    a state has code tables and counties
        StName, Census_Entities, School_District, Counties)>

<!ELEMENT Counties (County*)>                        a county has up to 14 records

<!ELEMENT County (
        CTY_1?, CTY_2?, CTY_3?, CTY_4?, CTY_5?, CTY_6?, CTY_7?, CTY_8?,
        CTY_a?, CTY_f?, CTY_g?, CTY_i?, CTY_p?, CTY_r?)>
```

Figure 6.3. Collection DTD for Tiger/Line collection

6.2 Ingestion process

Ingestion Statistics (real times @xena.sdsc.edu).

The ingestion of the Tiger/Line collection was done on a workstation, Xena.sdsc.edu. The system configuration was a Sun Microsystems sun4u Sun Ultra 2 UPA/Sbus (2X UltraSPARC 200MHz), System clock frequency: 100MHz, Memory size: 256 Megabytes

collection size:	24.5 Gbytes (on 44 CDs)
#files:	50,951
time to create tar archive:	19h 28m
time to store archive in HPSS:	18h 20m

Figure 6.4. Ingestion times for the Tiger/Line collection

The entire ingestion process was carried out manually over the course of a week. The average ingestion time per CD (600MB size) was:

tar:	**1592 sec.** (~27 min.)
HSI put, HPSS:	**1500 sec.** (~25 min.) **(=400Kbytes/sec)**

The TAR time is the time needed to concatenate all data for storage. The HIS put time is the time used to run the HSI utility to do the data storage. This I/O rate was limited by the network connecting the workstation to the HPSS system.

7. Collection support - 104th Congress

COLLECTION DESCRIPTION:

A collection of acts, bills, resolutions etc. of the 104th Congress of the United States

OBJECTIVES:

Demonstrate *preservation strategy* for a formatted text collection.

RELATED COLLECTIONS:

All other collections (other than ***JTIC***) have structured records, making this collection rather unique.

APPROACH TAKEN:

Automatic meta-data extraction is achieved by interpreting filenames. Documents need to be classified according to internal format similarity, so that classes of DTDs can be proposed. Appropriate tag names need to be created. This process is illustrated by looking at several example scenarios.

IMPORTANT FINDINGS:

The work is in progress, however, beyond *modeling the documents* in the collection, there is also a need to *model the relationships* between different documents. Devising meaningful classes of relationships would be an important finding.

7.1 Information model

Generation of DTD

Some data analysis and mining was performed in order to define a DTD for each type of document in the collection. The validation of this DTD will require the involvement of representatives from the appropriate agency to decide relevant document tag names for use in the DTD. To illustrate the feasibility of converting these proceedings into XML under a well-defined DTD, we give a few examples of the DTD conforming to some documents in the collection. The corresponding XML files are at then end of this section.

Example 1:

For an amendment (suffix "eas" or "eah") the DTD is given in Figure 7.1:

```
<!ELEMENT DOCUMENT (CONGRESS|RESOLUTION|ATTESTATION)* >
<!ATTLIST DOCUMENT DOCID CDATA #IMPLIED>
<!ATTLIST DOCUMENT REFERENCENUMBER CDATA #IMPLIED>
<!ATTLIST DOCUMENT CLASS CDATA #IMPLIED>

<!ELEMENT CONGRESS (NUMBER|SESSIONNUMBER|(BODYOFCONGRESS)*|DATE) >
         <!ELEMENT NUMBER (#PCDATA)* >
         <!ELEMENT SESSIONNUMBER (#PCDATA)* >
         <!ELEMENT BODYOFCONGRESS (#PCDATA)* >
         <!ELEMENT DATE (#PCDATA)* >
         <!ELEMENT RESOLUTION (TYPE|STATEMENT)* >
         <!ELEMENT TYPE (#PCDATA)* >
         <!ELEMENT STATEMENT (OPENING|(STRIKEOUT|INSERT)*|CLOSING) >
         <!ELEMENT OPENING (#PCDATA)* >
         <!ELEMENT STRIKEOUT (#PCDATA)* >
         <!ELEMENT INSERT (#PCDATA)* >
         <!ELEMENT CLOSING (#PCDATA)* >
         <!ELEMENT ATTESTATION (#PCDATA)* >
```

Figure 7.1 DTD for documents with suffix "eas" or "eah"

Example 2

A more involved example is an Act document (suffix "eh" or "es") which can be modeled by the structure given below in Figure 7.2.

```
<!ELEMENT DOCUMENT (CONGRESS|ACT)* >

<!ATTLIST DOCUMENT DOCID CDATA #IMPLIED>
<!ATTLIST DOCUMENT REFERENCENUMBER CDATA #IMPLIED>
<!ATTLIST DOCUMENT CLASS CDATA #IMPLIED>
```

```
<!ELEMENT CONGRESS (NUMBER|SESSIONNUMBER|BODYOFCONGRESS|DATE) >

         <!ELEMENT NUMBER (#PCDATA)* >
         <!ELEMENT SESSIONNUMBER (#PCDATA)* >
         <!ELEMENT BODYOFCONGRESS (#PCDATA)* >
         <!ELEMENT DATE (#PCDATA)* >

<!ELEMENT ACT (PURPOSE|SECTION)* >

         <!ELEMENT PURPOSE (#PCDATA)* >
         <!ELEMENT SECTION (HEADING|STATEMENT|SUBSECTION)* >
         <!ATTLIST SECTION NUMBER CDATA #IMPLIED>

         <!ELEMENT HEADING (#PCDATA|SHORTTITLE)* >
         <!ELEMENT SHORTTITLE (#PCDATA)* >

         <!ELEMENT STATEMENT (#PCDATA | AMENDMENT | SECTION |
         ATTESTATION | TEXT)* >
         <!ELEMENT AMENDMENT (CODE|STRIKEOUT|INSERT|REDESIGNATE)* >
         <!ELEMENT CODE (#PCDATA)* >
         <!ELEMENT STRIKEOUT (#PCDATA)* >
         <!ELEMENT INSERT (#PCDATA|PARAGRAPH|SUBSECTION|INSERT)* >
         <!ELEMENT PARAGRAPH (TOPIC|TEXT|PARAGRAPH)* >
         <!ATTLIST PARAGRAPH NUMBER CDATA #IMPLIED>

         <!ELEMENT SUBSECTION (TOPIC|PARAGRAPH)* >
         <!ATTLIST SUBSECTION NUMBER CDATA #IMPLIED>

         <!ELEMENT TOPIC (#PCDATA)* >
         <!ELEMENT TEXT (#PCDATA)* >
         <!ELEMENT REDESIGNATE (#PCDATA)* >
         <!ELEMENT ATTESTATION (#PCDATA)* >
```

Figure 7.2. DTD for an acts document

Note that since the act under consideration can be the modification of an existing act or can result in changes in existing acts, the DTD for amendments from Example 1 is in some sense "embedded" in Example 2. XML allows this "sharing" of smaller common DTDs by one or more larger DTDs through a notation called Entity. In the case of this example an Entity called Amendment can be specified separately and be declared in the Act DTD. After the declaration it can be used as:

```
<!ELEMENT STATEMENT (#PCDATA | AMENDMENT | SECTION | ATTESTATION |
TEXT)* >
```

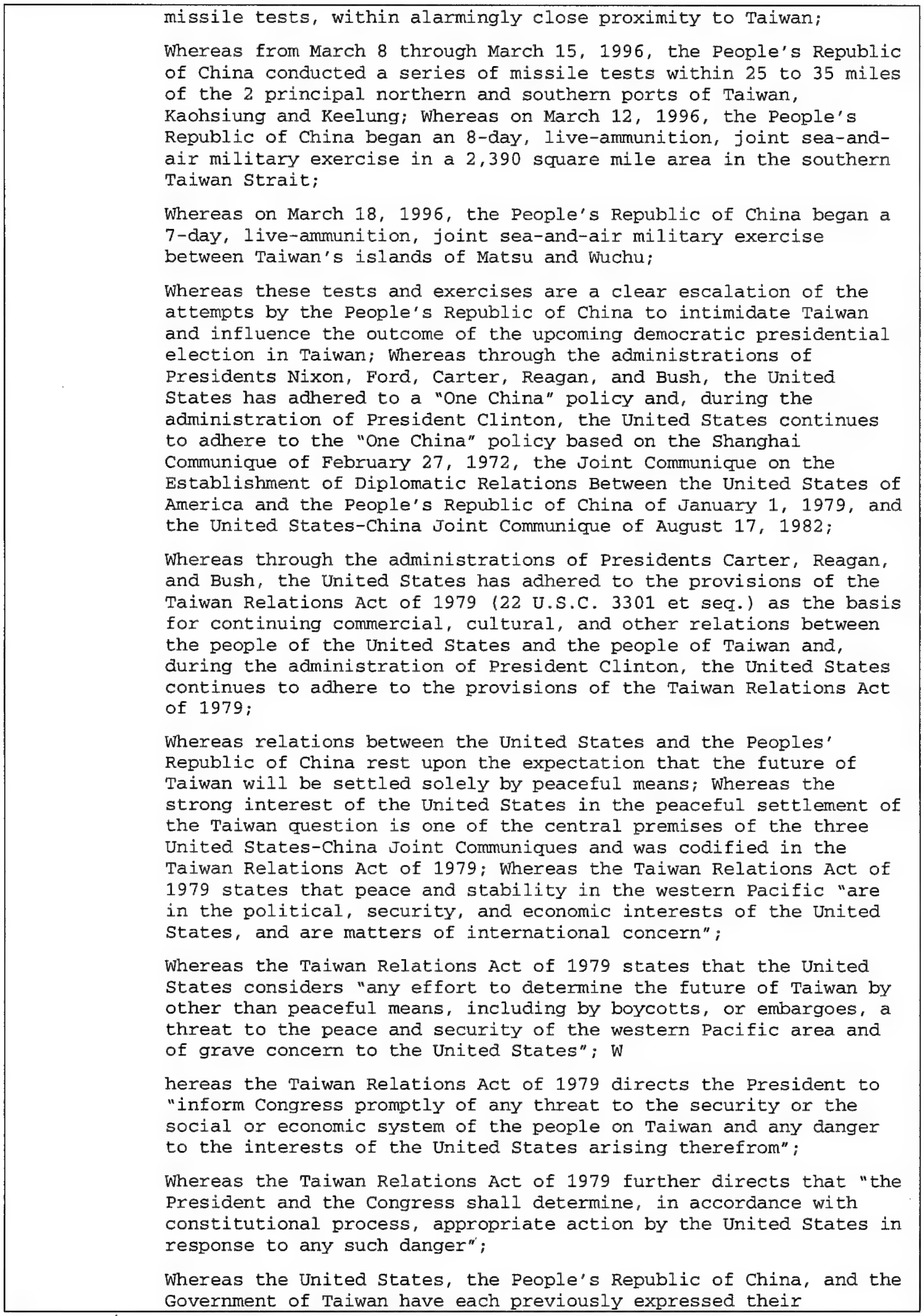
without having to expand AMENDMENT as done in the example DTD.

The XML version of an amendment is shown in Figure 7.3.

```
<DOCUMENT DOCID="f:hc148eas.txt" REFERENCENUMBER = "H. CON. RES. 148"
CLASS="Concurrent Resolution">

<CONGRESS>
   <NUMBER>104th CONGRESS</NUMBER>
   <SESSIONNUMBER>2d Session</SESSIONNUMBER>
   <BODYOFCONGRESS>Senate of the United States</BODYOFCONGRESS>
   <DATE>March 21, 1996</DATE>
</CONGRESS>

<RESOLUTION>
        <TYPE>AMENDMENTS</TYPE>
        <STATEMENT>
        <OPENING>
                Resolved, That the resolution from the House of Representatives
                (H. Con. Res. 148) entitled "Concurrent resolution expressing the
                sense of the Congress that the United States is committed to
                military stability in the Taiwan Strait and the United States
                should assist in defending the Republic of China (also known as
                Taiwan) in the event of invasion, missile attack, or blockade by
                the People's Republic of China.", do pass with the following
        </OPENING>
        <STRIKEOUT>Strike out all after the resolving clause
        </STRIKEOUT>
        <INSERT>
        That it is the sense of the Congress—
                (1)    to deplore the missile tests and military exercises that
                the People's Republic of China is conducting from March 8 through
                March 25, 1996, and view such tests and exercises as potentially
                serious threats to the peace, security, and stability of Taiwan
                and not in the spirit of the three United States-China Joint
                Communiques;
                (2)    to urge the Government of the People's Republic of China to
                cease its bellicose actions directed at Taiwan and enter instead
                into meaningful dialogue with the Government of Taiwan at the
                highest levels, such as through the Straits Exchange Foundation in
                Taiwan and the Association for Relations Across the Taiwan Strait
                in Beijing, with an eye towards decreasing tensions and resolving
                the issue of the future of Taiwan;
                (3)    that the President should, consistent with section 3© of
                the Taiwan Relations Act of 1979 (22 U.S.C. 3302©), immediately
                consult with Congress on an appropriate United States response to
                the tests and exercises should the tests or exercises pose an
                actual threat to the peace, security, and stability of Taiwan; (4)
                       that the President should, consistent with the Taiwan
                Relations Act of 1979 (22 U.S.C. 3301 et seq.), reexamine the
                nature and quantity of defense articles and services that may be
                necessary to enable Taiwan to maintain a sufficient self-defense
                capability in light of the heightened military threat; and
                (5)    that the Government of Taiwan should remain committed to
                the peaceful resolution of its future relations with the People's
                Republic of China by mutual decision.
        </INSERT>
        <STRIKEOUT>Strike out the preamble</STRIKEOUT>
        <INSERT>
                Whereas the People's Republic of China, in a clear attempt to
                intimidate the people and Government of Taiwan, has over the past
                9 months conducted a series of military exercises, including
```



missile tests, within alarmingly close proximity to Taiwan;

Whereas from March 8 through March 15, 1996, the People's Republic of China conducted a series of missile tests within 25 to 35 miles of the 2 principal northern and southern ports of Taiwan, Kaohsiung and Keelung; Whereas on March 12, 1996, the People's Republic of China began an 8-day, live-ammunition, joint sea-and-air military exercise in a 2,390 square mile area in the southern Taiwan Strait;

Whereas on March 18, 1996, the People's Republic of China began a 7-day, live-ammunition, joint sea-and-air military exercise between Taiwan's islands of Matsu and Wuchu;

Whereas these tests and exercises are a clear escalation of the attempts by the People's Republic of China to intimidate Taiwan and influence the outcome of the upcoming democratic presidential election in Taiwan; Whereas through the administrations of Presidents Nixon, Ford, Carter, Reagan, and Bush, the United States has adhered to a "One China" policy and, during the administration of President Clinton, the United States continues to adhere to the "One China" policy based on the Shanghai Communique of February 27, 1972, the Joint Communique on the Establishment of Diplomatic Relations Between the United States of America and the People's Republic of China of January 1, 1979, and the United States-China Joint Communique of August 17, 1982;

Whereas through the administrations of Presidents Carter, Reagan, and Bush, the United States has adhered to the provisions of the Taiwan Relations Act of 1979 (22 U.S.C. 3301 et seq.) as the basis for continuing commercial, cultural, and other relations between the people of the United States and the people of Taiwan and, during the administration of President Clinton, the United States continues to adhere to the provisions of the Taiwan Relations Act of 1979;

Whereas relations between the United States and the Peoples' Republic of China rest upon the expectation that the future of Taiwan will be settled solely by peaceful means; Whereas the strong interest of the United States in the peaceful settlement of the Taiwan question is one of the central premises of the three United States-China Joint Communiques and was codified in the Taiwan Relations Act of 1979; Whereas the Taiwan Relations Act of 1979 states that peace and stability in the western Pacific "are in the political, security, and economic interests of the United States, and are matters of international concern";

Whereas the Taiwan Relations Act of 1979 states that the United States considers "any effort to determine the future of Taiwan by other than peaceful means, including by boycotts, or embargoes, a threat to the peace and security of the western Pacific area and of grave concern to the United States"; W

hereas the Taiwan Relations Act of 1979 directs the President to "inform Congress promptly of any threat to the security or the social or economic system of the people on Taiwan and any danger to the interests of the United States arising therefrom";

Whereas the Taiwan Relations Act of 1979 further directs that "the President and the Congress shall determine, in accordance with constitutional process, appropriate action by the United States in response to any such danger";

Whereas the United States, the People's Republic of China, and the Government of Taiwan have each previously expressed their

```
             commitment to the resolution of the Taiwan question through
             peaceful means; and

             Whereas these missile tests and military exercises, and the
             accompanying statements made by the Government of the People's
             Republic of China, call into serious question the commitment of
             China to the peaceful resolution of the Taiwan question: Now,
             therefore, be it.

      </INSERT>
      <CLOSING>
             Amend the title so as to read: "Expressing the sense of
             Congress regarding missile tests and military exercises by the
             People's Republic of China.".
      </CLOSING>
      </STATEMENT>
</RESOLUTION>

<ATTESTATION>Secretary</ATTESTATION>

</DOCUMENT>
```

Figure 7.3. DTD for an amendment

Aside from modeling documents in the collections, we also need to model the relationships between different documents. The *foremost relationship* is among documents that bear the same document number (denoted as <number> above), but which have different text forms. Although each document is a distinct physical entity, they represent different logical stages that a bill goes through from its introduction, through various considerations to its final enactment or suspension of action. In the collection provided, documents Hc200ih.txt, Hc200rh.txt, Hc200eh.txt and Hc200rds.txt represent the steps through the *lifecycle of the bill*.

A *second class of relationships* is when one document in a collection refers to another document that may not be in the same collection. A typical example is a reference to a section in the Constitution or a prior Act proposed to be modified.

7.2 Ingestion process

We have converted the collection into a tar file and have placed the tar file in HPSS, our long-term persistent storage system. The time for the ingestion process is reported in Table 7.1

:

Size	317M bytes
Number of Files	11437
Time to convert to tar file	13m 57s
Time to store in HPSS	14m 56s

Table 7.1. Ingestion times for 104th Congress data

The data for this collection is digital, although it has been transcribed from paper documents. While the quality of the digital document does not degrade, we are unsure of the faithfulness of the digital records to the original. We have observed that the process of transcription from paper document to digital document sometimes creates a semantic ambiguity in the way a reference is made to the content of the document. For example, an amendment consists of insertions and deletions of text from an original document. The place of insertion (or deletion) is often referred by line number and page number. However, in the digital version of the document, page breaks are not marked, leading to potential information loss in establishing the correct reference.

7.3 Storage requirements

The size of the raw text files is 317 MB. The size of the tar file is 325.8 MB, a modest 3% increase.

8. Collection support - Vote Archive Demo 1997 (VAD97)

COLLECTION DESCRIPTION:

This collection contains U.S. House of Representatives Roll Call Votes for the 105th Congress - 1st Session (1997) as compiled through the electronic voting machine. The collection is very well structured with good documentation

OBJECTIVES:

Demonstrate *preservation strategy* for a mix of fixed-format and HTML pages.

RELATED COLLECTIONS:

This collection naturally relates to the ***104th*** collection. However, the format of these two collections is very different. One primarily deals with semi-structured text (HTML pages), the other with unstructured text (ASCII files) where structure needs to be discovered.

APPROACH TAKEN:

This collection has been ingested but not further processed. The structure of the associated text files and of the HTML pages is tabular, hence these files can be easily XML-tagged according to a well-defined DTD.

IMPORTANT FINDINGS:

An important finding is the need to preserve Web page links and the associated content to represent this collection. Further study is required on the limits to the amount and type of information one would want to preserve when dealing with Web page links.

8.1 Information model

The major issues relate to the validation of the data collection which contains duplicate data sets. For instance, the HTML index file contains not only pointers to the roll call votes, but also to a subset of the records from the last roll call file. The data sets as received can be archived. However it will be difficult to maintain the duplicate data records in a database representation of the collection. It is possible to add attributes to denote when duplicate data records are provided. The question is whether such information will be of value when the collection is archived.

8.2 Ingestion process

We have converted the collection into a tar file and have placed the tar file in HPSS, our long-term persistent storage system. The time for the ingestion process is reported in Table 8.1:

Size	33 Mbytes
Number of Files	1,288 files (640 rolls and votes)
Time to convert to tar file	3m 23s
Time to store in HPSS	2m 25s

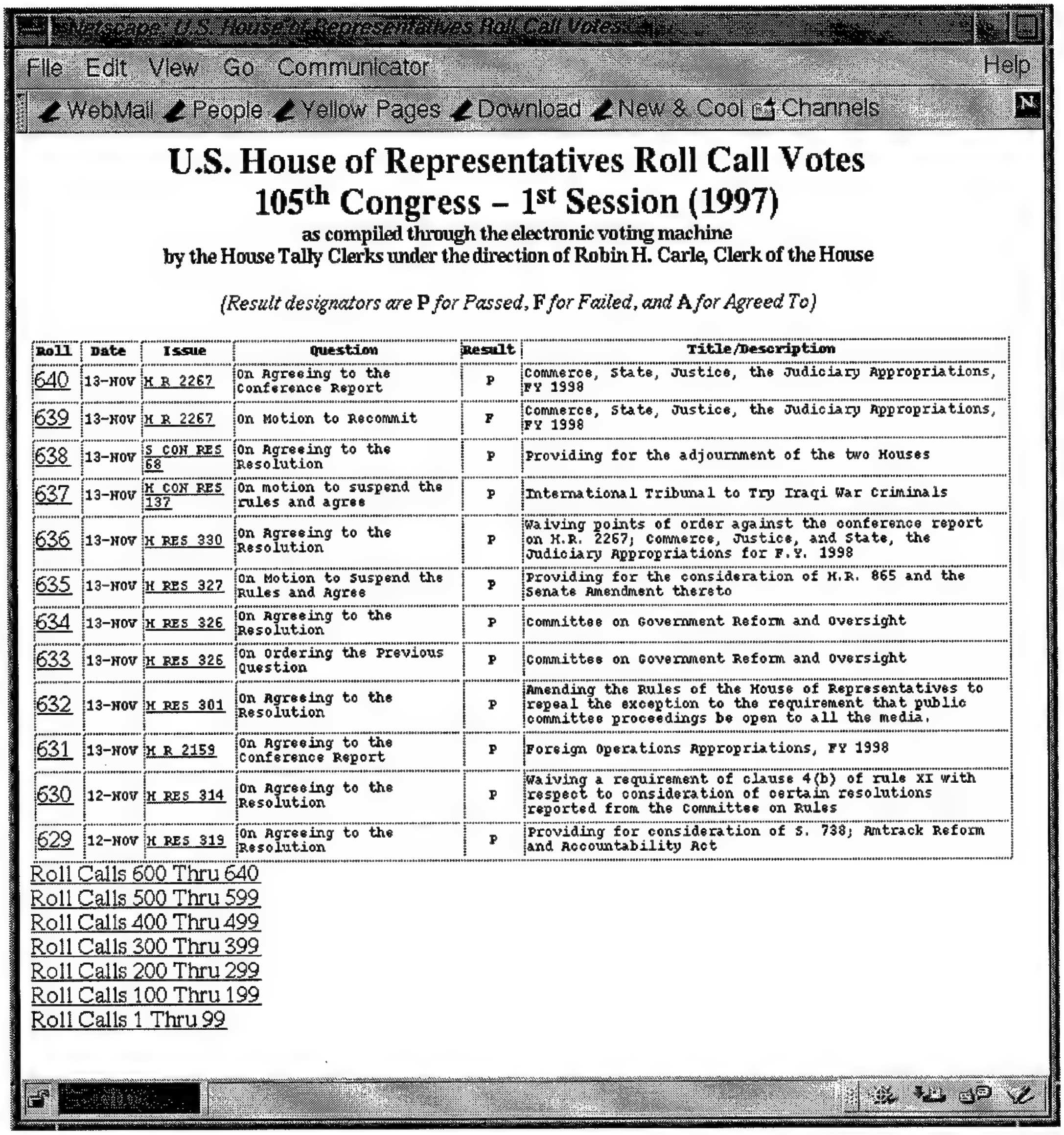
Table 8.1. Ingestion times for the Vote Archive

8.3 Long term preservation requirements

The following 3 screen snapshots illustrate some of the issues involved with this collection. The first screen is the main *index.htm* file. While the first column in the table points to a tabular document with a regular structure (see 2nd screen snapshot), the third column in the table points to a related Web site (see 3rd screen snapshot). Note that including the information that is linked over the Web requires accessing each Web site and retrieving additional data beyond that provided in the original tables.

Decisions on standards for representing the related but external links are needed, as well as policy decisions on how much or how little of this information needs to be preserved.

Major issues to be addressed are: *preservation of the Web* itself, and information discovery or *wrapping* of Web sites.



U.S. House of Representatives Roll Call Votes
105th Congress – 1st Session (1997)
as compiled through the electronic voting machine
by the House Tally Clerks under the direction of Robin H. Carle, Clerk of the House

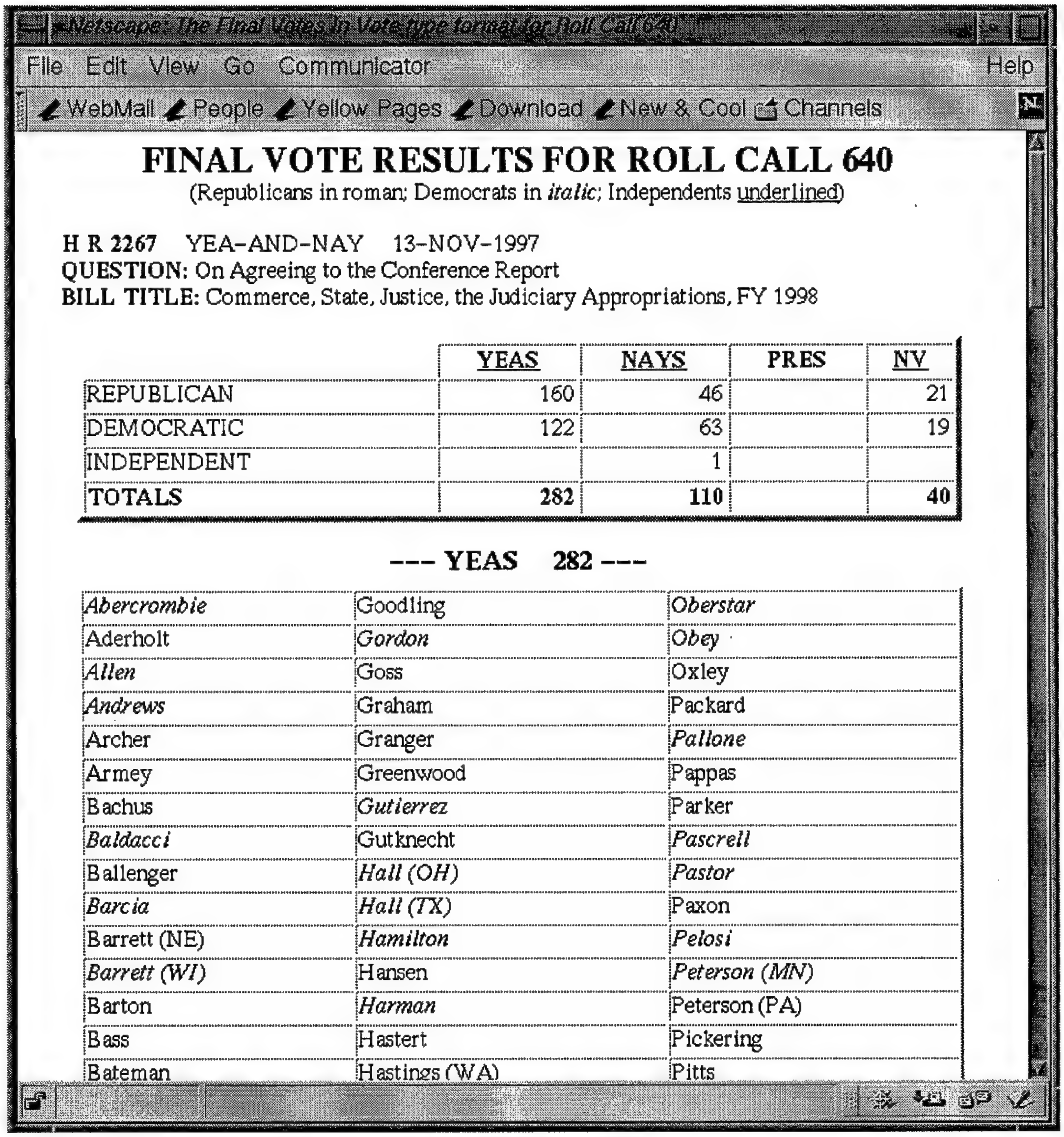
(Result designators are **P** *for Passed,* **F** *for Failed, and* **A** *for Agreed To)*

Roll	Date	Issue	Question	Result	Title/Description
640	13-NOV	H R 2267	On Agreeing to the Conference Report	P	Commerce, State, Justice, the Judiciary Appropriations, FY 1998
639	13-NOV	H R 2267	On Motion to Recommit	F	Commerce, State, Justice, the Judiciary Appropriations, FY 1998
638	13-NOV	S CON RES 68	On Agreeing to the Resolution	P	Providing for the adjournment of the two Houses
637	13-NOV	H CON RES 137	On motion to suspend the rules and agree	P	International Tribunal to Try Iraqi War Criminals
636	13-NOV	H RES 330	On Agreeing to the Resolution	P	Waiving points of order against the conference report on H.R. 2267; Commerce, Justice, and State, the Judiciary Appropriations for F.Y. 1998
635	13-NOV	H RES 327	On Motion to Suspend the Rules and Agree	P	Providing for the consideration of H.R. 865 and the Senate Amendment thereto
634	13-NOV	H RES 326	On Agreeing to the Resolution	P	Committee on Government Reform and Oversight
633	13-NOV	H RES 326	On Ordering the Previous Question	P	Committee on Government Reform and Oversight
632	13-NOV	H RES 301	On Agreeing to the Resolution	P	Amending the Rules of the House of Representatives to repeal the exception to the requirement that public committee proceedings be open to all the media.
631	13-NOV	H R 2159	On Agreeing to the Conference Report	P	Foreign Operations Appropriations, FY 1998
630	12-NOV	H RES 314	On Agreeing to the Resolution	P	Waiving a requirement of clause 4(b) of rule XI with respect to consideration of certain resolutions reported from the Committee on Rules
629	12-NOV	H RES 319	On Agreeing to the Resolution	P	Providing for consideration of S. 738; Amtrack Reform and Accountability Act

Roll Calls 600 Thru 640
Roll Calls 500 Thru 599
Roll Calls 400 Thru 499
Roll Calls 300 Thru 399
Roll Calls 200 Thru 299
Roll Calls 100 Thru 199
Roll Calls 1 Thru 99

Figure 8.1. First Screen Snapshot of the main index file

File *index.htm* when viewed in a web browser is actually the above master index document.



FINAL VOTE RESULTS FOR ROLL CALL 640

(Republicans in roman; Democrats in *italic*; Independents underlined)

H R 2267 YEA-AND-NAY 13-NOV-1997
QUESTION: On Agreeing to the Conference Report
BILL TITLE: Commerce, State, Justice, the Judiciary Appropriations, FY 1998

	YEAS	NAYS	PRES	NV
REPUBLICAN	160	46		21
DEMOCRATIC	122	63		19
INDEPENDENT		1		
TOTALS	**282**	**110**		**40**

--- YEAS 282 ---

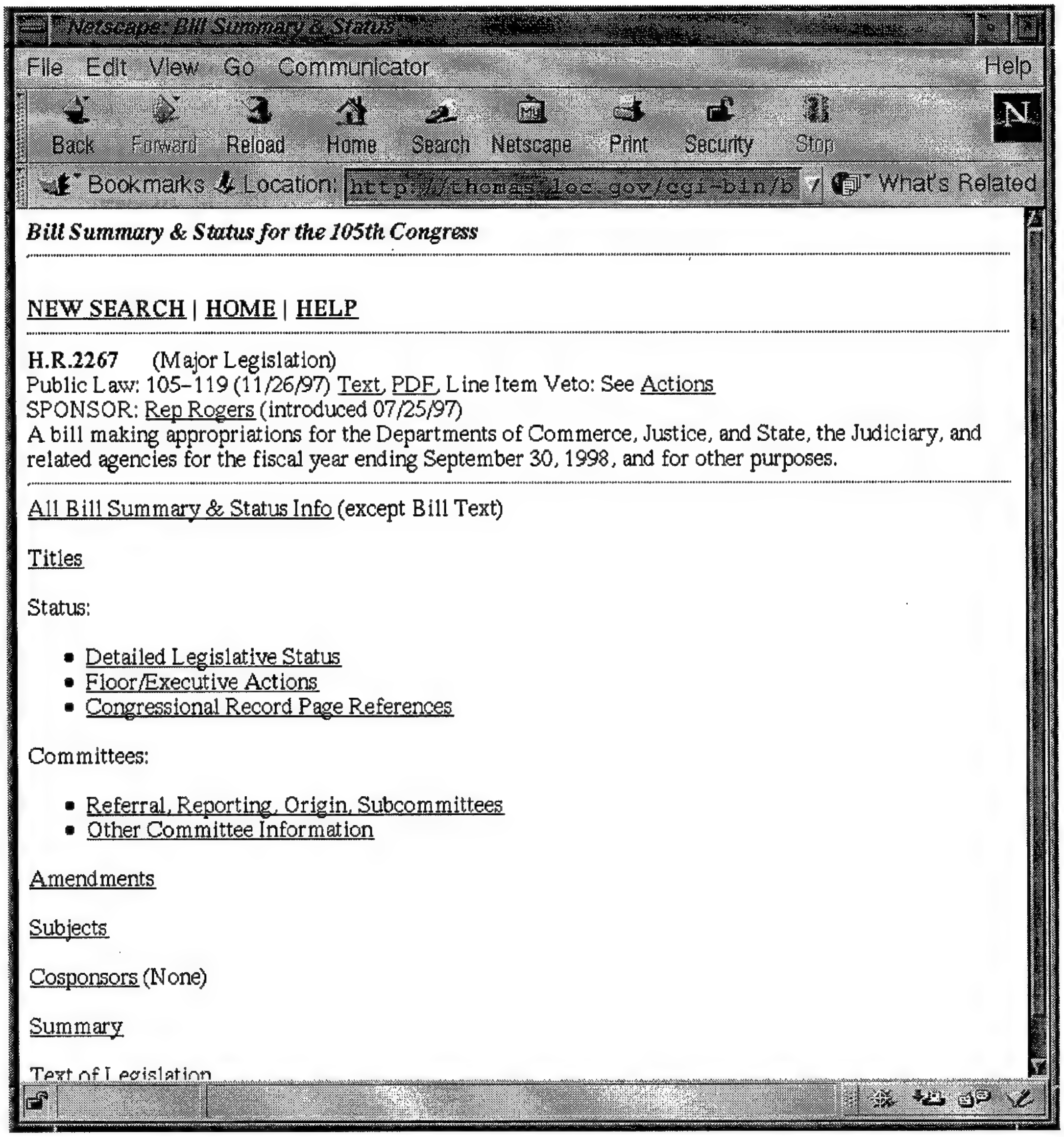
Abercrombie
Aderholt
Allen
Andrews
Archer
Armey
Bachus
Baldacci
Ballenger
Barcia
Barrett (NE)
Barrett (WI)
Barton
Bass
Bateman

Goodling
Gordon
Goss
Graham
Granger
Greenwood
Gutierrez
Gutknecht
Hall (OH)
Hall (TX)
Hamilton
Hansen
Harman
Hastert
Hastings (WA)

Oberstar
Obey
Oxley
Packard
Pallone
Pappas
Parker
Pascrell
Pastor
Paxon
Pelosi
Peterson (MN)
Peterson (PA)
Pickering
Pitts

Figure 8-2. Second screen snapshot of the roll 640 link

This screen snapshot was obtained after following the **roll 640** Web link.


Netscape: Bill Summary & Status
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://thomas.loc.gov/cgi-bin/b
What's Related

Bill Summary & Status for the 105th Congress

NEW SEARCH | HOME | HELP

H.R.2267 (Major Legislation)
Public Law: 105-119 (11/26/97) Text, PDF, Line Item Veto: See Actions
SPONSOR: Rep Rogers (introduced 07/25/97)
A bill making appropriations for the Departments of Commerce, Justice, and State, the Judiciary, and related agencies for the fiscal year ending September 30, 1998, and for other purposes.

All Bill Summary & Status Info (except Bill Text)

Titles

Status:

- Detailed Legislative Status
- Floor/Executive Actions
- Congressional Record Page References

Committees:

- Referral, Reporting, Origin, Subcommittees
- Other Committee Information

Amendments

Subjects

Cosponsors (None)

Summary

Text of Legislation

Figure 8.3. Third screen snapshot of Web linked data

This screen corresponds to the Issue **H R 2267** linked field of Roll 640.

9. Collection support – Electronic Archiving Project (EAP)

COLLECTION DESCRIPTION:

The collection consists of a subset of the images in *NAIL* (NARA Archival Information Locator) plus the *NAIL* lookup table with associated meta-data. This collection was the most difficult to manage, as the digital objects and meta-data were provided on three different types of media. Also, the meta-data is basically the output from a program, and is not structured via a standard format.

OBJECTIVES:

Demonstrate *preservation strategy* for a collection of images, using a proprietary database format.

RELATED COLLECTIONS:

This collection is similar to the ***AMICO*** digital image collection. The techniques and tools demonstrated for the AMICO collection should also apply to ***EAP***.

APPROACH TAKEN:

While ingestion of the data was performed and while we attempted to discover the underlying image meta-data, we focused instead on a similar image collection. Please refer to ***AMICO***.

IMPORTANT FINDINGS:

Please see the ***AMICO*** section.

9.1 Information model

The NAIL lookup table contains all meta-data for the individual objects. However, the table is in a proprietary format for which we do not have documentation. A preliminary analysis revealed a RECORD STRUCTURE, where each record contains a sequence of ATTRIBUTE/VALUE PAIRS.

There seem to be around 80-100 attribute names. Their meaning can be partially (and painfully) "reverse-engineered" by matching (meta)data entries on the NAIL Web site with corresponding ones in the NAIL lookup table (see the EAP subdirectory described in Appendix E).

MISCELLANEOUS:

Ingestion of raw data and some preliminary analysis of the proprietary format were straightforward; extraction of meta-data at the object level will be difficult, especially if no documentation or tools for parsing the lookup table are available.

9.2 Ingestion process

So far we have converted the collection into a tar file and have placed the tar file in HPSS, our long-term persistent storage system. The time for the ingestion process is reported below:

Size	861 Mbytes
Number of Files	11,543
Time to convert to tar file	41m 37s
Time to store in HPSS	37m 13s

10. Collection support - Combat Area Casualties Current File (CACCF)

COLLECTION DESCRIPTION:

The collection contains the Vietnam era Combat Area Casualties Files (CACCF), a list of Vietnam War casualties from 1957 to 1986, compiled by the Secretary of Defense.

OBJECTIVES:

Demonstrate *preservation strategy* for a fixed-record format collection.

RELATED COLLECTIONS:

Other highly structured collections are similar, such as ***Tiger92*** and ***104th***.

APPROACH TAKEN:

Due to the highly regular structure, a relational database schema and an XML DTD are built for this collection. Ad-hoc SQL queries are generated as well as XML-queries using the XMAS query language. Inconsistencies in the data are detected upon the loading of the records.

IMPORTANT FINDINGS:

Because the data can be represented as a relational database, a variety of consistency checks can be performed using the built-in mechanisms provided by such database systems (such as counts by casualty country, etc.).

10.1 Information model

Collection Level Structure.

The structure at the *collection level* can be described by the following XML DTD:

```
<!ELEMENT caccf_collection (caccf_database)*>          list of CACCF databases

<!ELEMENT caccf_database (                             a single database
                        date_id,                       creation date, identifies the database
                        record_size,                   size of each record (bytes)
                        caccf_records                  the actual data records
)>

<!ELEMENT caccf_records (
                        caccf_record*                  list of caccf  records
)>
```

Figure 10.1. DTD for the CACCF collection

Object Level Structure: XML DTD.

The structure and meaning of the individual data objects (records) can be obtained from the hardcopy documentation, which describes the format of the most recent versions (i.e., where *record_size* = 164 bytes). For these a DTD was derived which represents every record field as an attribute (alternatively, one could use subelements instead of attributes). The DTD is shown in Figure 10.2.

```
<!ELEMENT caccf_record EMPTY          all info is in the attributes
)>
<!ATTLIST caccf_record                length  meaning
        ms      CDATA #REQUIRED       1       Military Service (DoD Component)
        cc      CDATA #REQUIRED       2       Country of Casualty
        tc      CDATA #REQUIRED       2       Type of Casualty
        rn      CDATA #REQUIRED       5       Reference Number (File Ref. No)
        na      CDATA #REQUIRED       28      Name (of Casualty)
        dp      CDATA #REQUIRED       4       Date Record Processed (YYMM)
        sn      CDATA #REQUIRED       9       Social Security OR Service Num.
        mg      CDATA #REQUIRED       4       Military Grade (Grade or Rate)
        pg      CDATA #REQUIRED       2       Pay Grade (Grade or Rate)
        dd      CDATA #REQUIRED       8       Date of Death (MM/DD/YY) (Casualty)
        hc      CDATA #REQUIRED       20      "Home of Record" City (Place)
        hs      CDATA #REQUIRED       2       "Home of Record" State Code
        oc      CDATA #REQUIRED       5       Service Occupation Code
        db      CDATA #REQUIRED       8       Date of Birth (MM/DD/YY)
        rc      CDATA #REQUIRED       1       Reason (Cause of Casualty)
        ai      CDATA #REQUIRED       1       Aircraft or Not Aircraft
        ra      CDATA #REQUIRED       1       Race
        re      CDATA #REQUIRED       2       Religion Code (Religious Denom.)
```

```
      le     CDATA #REQUIRED    2     Length of Service in Years
      ma     CDATA #REQUIRED    1     Marital Status
      se     CDATA #REQUIRED    1     Sex
      ci     CDATA #REQUIRED    1     Citizen Code
      pp     CDATA #REQUIRED    1     Posthumous Promotion
      dt     CDATA #REQUIRED    6     Date Tour in Southeast Asia
      lr     CDATA #REQUIRED    1     Last Record Code
      br     CDATA #REQUIRED    3     Body Recovered or Not
      ag     CDATA #REQUIRED    2     Age at Time of Casualty
      sc     CDATA #REQUIRED    1     Component (Service Component)
      co     CDATA #REQUIRED    29    Comments
      ty     CDATA #REQUIRED    2     Type
      pc     CDATA #REQUIRED    2     Province Code (South Vietnam Provinces &
                                       Military Regions)
      mc     CDATA #REQUIRED    2     CORPCD
      pr     CDATA #REQUIRED    2     PROCD
      fl     CDATA #REQUIRED    2     Flag
)>
```

Figure 10.2. DTD for CACCF data objects

Object Level Structure: Relational Schema.

Loading sources with regular structure into a relational database (RDB) has several benefits as can be seen from the CACCF case (cf. below):

- Inconsistencies in the data can be automatically detected using the RDB's built-in *consistency checks* (data types, uniqueness of keys, referential integrity, etc.)
- Powerful ad-hoc SQL queries can be used to further *clean* the data from inconsistencies
- Interesting information from the collection can be *mined*
- Different versions of the collection can be *compared*
- Using RDB-to-XML wrappers, after loading the data into the RDB, the RDB-to-XML wrapper provides an XML view on the collection

For loading the CACCF into a relational database (here: Oracle), the following schema was used:

```
create table CACCF (
   REC_NO           int not null, -- DICE: no. as found in the source
   MIL_SERVICE      char (1),     -- Military Service (DoD Component)
   CASUALTY_COUNTRY char (2),     -- Country of Casualty
   CASUALTY_TYPE    char (2),     -- Type of Casualty
   REF_NO           char (5),     -- Reference Number (File Ref. No)
   NAME             char (28),    -- Name (of Casualty)
   PROCESSED        char (4),     -- Date Record Processed (YYMM)
   SSN_SERVICE_NO   char (9),     -- Social Security OR Service Num.
   GRADE            char (4),     -- Military Grade (Grade or Rate)
   PAY_GRADE        char (2),     -- Pay Grade (Grade or Rate)
   DIED             date,         -- Date of Death (MM/DD/YY)(Casualty)
   HOR_CITY         char (20),    -- "Home of Record" City (Place)
   HOR_STATE        char (2),     -- "Home of Record" State Code
   OCCUPATION       char (5),     -- Service Occupation Code
   BORN             date,         -- Date of Birth (MM/DD/YY)
```

```
    CASUALTY_REASON  char (1),      -- Reason (Cause of Casualty)
    AIR              char (1),      -- Aircraft or Not Aircraft
RACE             char (1),     -- Race
RELIGION         char (2),     -- Religion Code (Religious Denom.)
SERVICE_LENGTH   char (2),     -- Length of Service in Years
MARITAL_STATUS   char (1),     -- Marital Status
SEX              char (1),     -- Sex
CITIZEN          char (1),     -- Citizen Code
PH_PROMOTION     char (1),     -- Posthumous Promotion
SEA_TOUR         date,         -- Date Tour in Southeast Asia
LAST_RECORD      char (1),     -- Last Record Code
BODY_RECOVERED   char (3),     -- Body Recovered or Not
AGE              char (2),     -- Age at Time of Casualty
COMPONENT        char (1),     -- Component (Service Component)
COMMENTS         char (29),    -- Comments
TYPE             char (2),     -- Type
PROVINCE         char (2),     -- Province Code (South Vietnam Provinces and
Military Regions)
CORPCD           char (2),     -- CORPCD
PROCD            char (2),     -- PROCD
FLAG             char (2),     -- Flag
NOTES            varchar (1000)  -- DICE: notes/corrections made
PRIMARY KEY      (SSN_SERVICE_NO),
UNIQUE           (REC_NO)
)
```

Figure 10.3. Schema for CACCF collection

The first and last attributes (marked with "DICE") were added to record the changes made when loading the data into the database.

10.2 Ingestion process

When reading the EBCDIC encoded files from tape, they were converted to ASCII (using the UNIX command "dd" for converting and copying files) and temporarily stored on disk. A Perl script was used to create the XML version (caccf2xml) and the Oracle version (caccf2oracle) of the data collection from the raw files. The consistency checks enforced by the relational database revealed a number of problems with the CACCF database (see below).

Ingestion Statistics (real times @xena.sdsc.edu).
Ingestion was done on a Sun workstation, with the system configuration: Sun Microsystems sun4u Sun Ultra 2 UPA/Sbus (2X UltraSPARC 200MHz), System clock frequency: 100MHz, Memory size: 256 Megabytes.

collection size:	75,740,672 bytes
#files:	8 (different versions of the CACCF database)
#records/file:	between 58,152 and 58,181
time to create tar archive:	2m 41s

time to store archive in HPSS:	2m 39s
conversion to XML:	1m per CACCF database (Perl script @xena.sdsc.edu)

Figure 10.4. Ingestion times for the CACCF collection

10.3 Storage requirements

The XML representation with the DTD shown in Figure 10.2 increases the file size by a factor of 2 or more. This is due to the fact that there are many short fields per record, each of which has to be marked. For example, using the XML DTD in Figure 10.2 the version RG330.CAC.C951129 requires 22,444,477 bytes vs. 9,540,700 bytes in the given format (size expansion ratio = 2.35:1).

By creating the XML data from the raw data *on the fly*, the overhead in storage can be completely avoided.

Inconsistencies in the Data.
Several SQL queries were executed and revealed that the social security number was NOT UNIQUE (two distinct entries with the same SSN/service number), and that several DATES WERE INCORRECT (Jun/Nov 31st etc.)

These inconsistencies were reported by the database system due to the use of the key constraint on SSN and the date format. The changes made when loading the data into the database are documented with the data and can be retrieved as follows:

```
SQL> select REC_NO,NAME, NOTES from CACCF where NOTES is not null;
54022 MC DANIEL JOHN THOMAS
SSN_SERVICE_NO was not unique; added "-?"
6809 MICHAEL DON LESLIE
changed 31-JUN-1947 to 30-JUN-1947
[...]
15 rows selected.
```

Figure 10.5. Select statements for retrieving inconsistent data

Detecting Incomplete Information.
The database can also be used to identify incomplete records for specific attributes.

```
/* Find records with incomplete BORN or DIED date: */
SQL> select REC_NO, NAME, BORN, DIED, AGE from CACCF
where  BORN is null or DIED is null;
REC_NO NAME                     BORN      DIED      AG
---------- ---------------------------- --------- --------- --
      2114 SEVENBERGEN JERRY L                    16-MAR-66 0
      3882 DOMINGUEZ MICHAEL J                    10-SEP-66 0
      3883 JORDAN ALLAN H                         10-SEP-66 0
...
33997 HALIBURTON MICHAEL R                08-AUG-70 0
```

```
    17 rows selected.
    /* How many entries do NOT have a value for 'Date Tour in Southeast
Asia'? */
SQL> select count (*) from CACCF where SEA_TOUR is null;
COUNT(*)

8241
```

Figure 10.6. Select statements for finding incomplete data

Detecting Possible Errors

The distribution of values for a certain attribute can be obtained by a query of the form

```
select count(*), <ATTRIBUTE>
from CACCF group by <ATTRIBUTE>
order by count(*) desc;
```

For example,

```
SQL> select count(*), SERVICE_LENGTH from CACCF
group by SERVICE_LENGTH
order by count(*) desc;
```

reveals inconsistencies in the SERVICE_LENGTH attribute (e.g. 3 different representations of presumably the same value "four", i.e. "O4", "04", and "4"):

Value-Added Data Retrieval.

If data is maintained in a structured form in a database (relational, object-oriented, or XML), much more flexible retrieval mechanisms can be provided to the user of the archive. As an example, consider the following "data mining/analysis" queries, which can be directly executed against the collection:

/* What is the number and min/max/average age of the casualties over entries where AGE is available? */

SQL> select count(*), min(AGE), max(AGE), avg(AGE)

from CACCF
where not AGE = '0';

```
COUNT(*)    MI    MA    AVG(AGE)
----------  --    --    ----------
   58164    16    62    22.7932742
```

The result of the query shows that minimum age was 16 years and the maximum was 62 years, with an average age of 22.79 years, out of a total number of casualties equal to 58,164.

```
/* Show The Distribution Of Casualties by  Country */

SQL> select count(*), CASUALTY_COUNTRY

from CACCF group by CASUALTY_COUNTRY order by count(*) desc;
```

```
COUNT(*)     CA
----------   --
  55621      VS
   1124      VN
    729      LA
    520      CB
    177      TH
     10      CH
```

The result lists the number of casualties by country acronym. Interpretation of the acronym requires an associated data dictionary. These are just a few examples which show how an RDB can yield "value-added" data by means of a flexible query language. Similar techniques can be applied to less regular data, provided a corresponding query language is available (e.g., XML data can be queried with the XML Matching And Structuring Query Language, XMAS [9]).

11. Collection support - Patent Data (USPTO)

COLLECTION DESCRIPTION:

A corpus of about 2 million U.S. patent text documents in a proprietary markup format, called the Greenbook format. Documents cover the time period beginning in 1971 until 1998. Documents were provided on 3480 tapes in the format generated by the USPTO where each tape contains all the patents issued in a given week. A total of about 1800 tapes were read. On average, each tape contains about 100MB of data and the total amount of data in bytes is about 150GB.

OBJECTIVES:

Demonstrate the capacity to *ingest*, *validate*, *archive,* and provide *access* to a large collection of archival text documents. Activities include conversion from legacy format to a standards-based formats and enumeration of any errors and inconsistencies encountered in the conversion process. Documents are also checked for correctness and consistency using other approaches including use of data analysis techniques and consistency checking during database load.

RELATED COLLECTIONS:

This is a collection of text documents which contain document markup. Related collections include the E-mail collection, which is a collection of text documents without markup, but where the markup can be inferred by processing the document using appropriate scripts.

APPROACH TAKEN:

The corpus of patent text documents was converted based on an SGML DTD as part of the DOCT project. An XML DTD is being defined for a second conversion of the collection. This conversion is planned for August, 1999.

IMPORTANT FINDINGS:

For a legacy data collection spanning a period of over 20 years, we found the collection to be in very good shape in terms of correctness and consistency of the documents. During the conversion from legacy to SGML format, only about 50 documents raised an error flag. In most cases, the documents were correct but the conversion script needed to be fixed. Only a handful of documents had errors most of which were easy to fix. Less than 5 remained which needed more work.

The claims graph analysis detected anomalies in about 1% of all the patents. A sample of these patents indicated that most of these are caused due to typographical errors. More important is the idea that such checking can be performed at the source, when the documents are originally submitted to the agency, thereby removing this possible source of error.

It was found that storing the "raw" data (i.e. the SGML files) as well as the data loaded into a relational database was useful to support long term preservation and immediate access. This was done using an advanced version of IBM's DB2 database system which is able to store large object data directly into the HPSS archive, rather than on disk.

11.1 PATENT DATA: FURTHER DETAILS

The corpus of patent text documents was ingested, validated, and archived using a multi-step process as described below:

(1) Reading of 3480 tapes directly into the HPSS archive at SDSC. The 3480 tapes received from the USPTO were read directly into the HPSS system at SDSC. Each tape corresponds to a single file in HPSS. Each such file contains the document text for all patents filed during a week. Documents are marked up using markup tags defined by the USPTO's proprietary Greenbook standard.

Tapes were received in batches, and were read into the HPSS system by the Operations staff at SDSC.

(2) Conversion of Greenbook files into SGML, based on a standard DTD. The USPTO uses an SGML DTD for describing patent documents. This DTD is based on an international standard DTD from the World Intellectual Property Organization (WIPO). Scripts were written to convert each Greenbook patent to an SGML document conforming to the specified DTD. In the process, any errors or inconsistencies in the document were recorded. In addition, the scripts were updates as required, as they encountered new aspects of the patent documents which were not originally accounted for (e.g. appearance of a subscript or superscript in the title of a document).

Some of the so-called errors that were flagged by the conversion process were due to shortcomings of the conversion script, which were fixed as they were discovered. Other errors were due to anomalies in some documents which were immediately addressed by the USPTO. At the end, there were only about 10 documents (out of about 2 million), which required further attention for corrections.

The entire conversion process for converting about 1200 Greenbook files containing about 2 million patents and 150GB of data, took about 40 hours using 15 parallel streams on a parallel Sun Enterprise Server computer (i.e. a total of about 600 hours)

(3) Storage of converted SGML files in HPSS. The converted SGML documents where then stored back as files (one per week, as in the original Greenbook) in HPSS.

(4) Design of a relational database schema based on Greenbook information and SGML DTD. Given the Greenbook specification and the SGML DTD, a relational schema was designed for storing documents efficiently. The schema was normalized and

contained one master table with individual tables for each of the sub-sections of the patent document, e.g. authors, attorney information, abstract, claims section, classificiation information, etc. In some cases, several tables are required for a sub-section due to the complex, nested nature of that sub-section.

The normalized schema contains about 40 relational tables.

(5) Loading of documents into relational database using appropriate mapping of data from the document into the relational schema. Once the relational schema was defined, database load scripts were written to load the patent document data into relational tables.

The total elapsed time for loading data into all the 40 tables in the database is about 4 days.

(6) Analysis of data in relational database to extract *claims graphs* for each patent and analysis of these graphs for consistency checking. A program was written to process each patent and extract the *claims graph* for the patent. The claims graph shows the dependencies among claims within a given patent. Once the graph is constructed, one can perform a variety of checks on the graphs to verify the consistency of the data. For example, one can check for cycles in the graph (there should be none)

The graph analysis program took a total elapsed time of 26 hours.

12. Collection support - Image collection (AMICO)

COLLECTION DESCRIPTION:

The AMICO image collection consists of a sample of high resolution images of art pieces acquired from the Art Museum Image Consortium (or, AMICO).

OBJECTIVES:

Demonstrate *preservation strategy* and *sustained access* to an image data collection, along with associated meta-data.

RELATED COLLECTIONS:

This collection is strongly similar to the ***EAP*** collection. The meta-data portion is similar to the ***Patent*** collection.

APPROACH TAKEN:

Meta-data was provided in proprietary markup format. A DTD was designed for the meta-data, based on the data dictionary provided by AMICO. The meta-data files were then converted into XML documents. Images are stored as binary objects.

IMPORTANT FINDINGS:

This work may lead to the formulation of key markup elements useful across all image collections.

12.1 Information model

Members of AMICO have established meta-data standards for describing art objects as well as the high resolution images of the art objects. The archiving process consists of, first, specifying an XML DTD to capture all the meta-data. This DTD is based on the data dictionary standard created by the AMICO consortium. Next, all the meta-data records provided by AMICO are converted into an XML form based on the particular DTD. Finally, the meta-data is archived along with the corresponding high-resolution images.

Organizing the Collection level meta-data

The AMICO data dictionary was provided to us in its original form as a Microsoft Excel spreadsheet. This is available at

http://www.npaci.edu/DICE/AMICO/Demo/amico-dd.1.2.xls.

The data dictionary includes a meta-data specification for the art objects as well as the corresponding digital objects, i.e. the images associated with the art objects. Thus, there are two classes of meta-data for each art object. Multiple images can be taken of a given art object. In addition, thumbnail image representations can be generated for each art object. Thus, an art object can be associated with one or more digital image objects. For ease of querying, archiving, and retrieving, a *single* meta-data object is associated with each art object. This meta-data object contains the art object (or, *artifact*) meta-data and the image object (or, *media*) meta-data—for each associated image. The archival object is composed of the meta-data object along with the set of associated image objects.

OBJECT LEVEL STRUCTURE/META-DATA

An XML DTD was designed for the meta-data object. Additionally, the DTD is designed to group the meta-data elements into an additional abstract classification, such as, *"What is it", "What is it called", "Who made it"*, based on the classification employed by AMICO itself. This meta-data DTD is available in amico-2in1.DTD.

A conversion program then translates the AMICO data dictionary records into XML meta-data documents that conform to the meta-data DTD.

Demo 1. Deriving the AMICO schema:

Demo 1.1. A first DTD, accessible at

http://www.npaci.edu/DICE/AMICO/Demo/amico-from-perl-DTD.txt ,

was obtained from the given AMICO data dictionary (which was provided as a spreadsheet), using a Perl script.

The spreadsheet is accessible at

http://www.npaci.edu/DICE/AMICO/Demo/amico-dd.1.2.xls

The Perl script is accessible at

http://www.npaci.edu/DICE/AMICO/Demo/amico-dd2dtd

Demo 1.2. This DTD was rewritten manually into the final AMICO DTD. The final DGD is available at

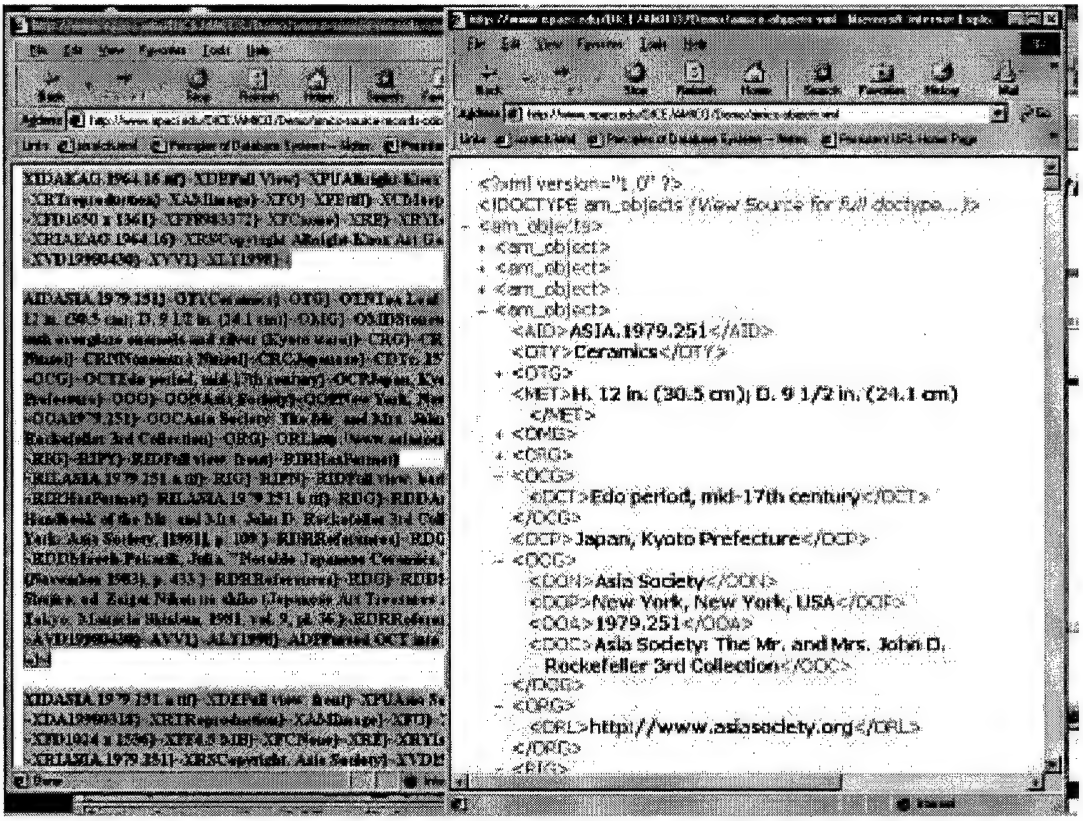
http://www.npaci.edu/DICE/AMICO/Demo/amico-2in1-DTD.txt

Demo 2. Conversion to XML:

The incoming raw data records were converted to XML using a Perl script, available at

http://www.npaci.edu/DICE/AMICO/Demo/amico2xml

Fig. 12-1 shows the input and output at this stage (left side: incoming raw data records; right side: records in XML format).



```
<?xml version="1.0" ?>
<!DOCTYPE am_objects (View Source for full doctype...)>
- <am_objects>
  + <am_object>
  + <am_object>
  + <am_object>
  - <am_object>
      <AID>ASIA.1979.251</AID>
      <OTY>Ceramics</OTY>
    + <OTG>
      <MET>H. 12 in. (30.5 cm); D. 9 1/2 in. (24.1 cm)
        </MET>
    + <OMG>
    + <CRG>
    - <OCG>
        <OCT>Edo period, mid-17th century</OCT>
      </OCG>
      <OCP>Japan, Kyoto Prefecture</OCP>
    - <OOG>
        <OON>Asia Society</OON>
        <OOP>New York, New York, USA</OOP>
        <OOA>1979.251</OOA>
        <OOC>Asia Society: The Mr. and Mrs. John D.
          Rockefeller 3rd Collection</OOC>
      </OOG>
    - <ORG>
        <ORL>http://www.asiasociety.org</ORL>
      </ORG>
    - <RIG>
```

Figure 12-1: Raw data records (left) and tagged XML (right)

12.2 Ingestion process

The result of the ingestion process is the storage of the entire AMICO collection in an archival storage system. Efficient access to the meta-data and image objects is provided using scaleable database and archival storage technology. The meta-data can be queried and accessed in the form of XML documents using the state of the art database technology. Currently, the best technology for providing a large number of users efficient concurrent access to large scale databases is, arguably, the relational database technology. Thus, the meta-data, in XML form, is converted into corresponding relational tables so that the entire collection is efficiently searchable using a relational database management system (RDBMS).

12.3 Data access requirements

Relational Storage Technology

For experimenting with relational technology, the AMICO meta-data was implemented as a relational database. For this purpose, a normalized relational data dictionary was designed, and the sample data was loaded into the SRB/MCAT system [5,6]. The data dictionary is available at

http://www.npaci.edu/DICE/AMICO/Demo/amico.ddl

DEMO INFO

Online versions of the AMICO demo are available at:

- AMICO/XML demo: http://www.npaci.edu/DICE/AMICO/Demo/
- AMICO/SRB demo: http://srb.npaci.edu/

The sample data consists of

- images (thumbnails and full images),
- an associated meta-data file, and
- the AMICO data dictionary.

There are two kinds of AMICO meta-data: the **amico-objects** meta-data (Fig.12-1, top and bottom sections of the left-hand side) contains meta-data about the artifact (title, creator, creation time, etc.), whereas the **amico-media** meta-data (Fig.12-1, middle section of the left-hand side) contains meta-data about the digital image (resolution, size, compression, etc.)

The AMICO demo illustrates various steps for archival storage of image collections and their associated meta-data.

Demo 3. Querying XML.

The tagged AMICO data can be queried using an XML query language. For the demo, we used the XMAS (*XML Matching And Structuring*) language.

Example.

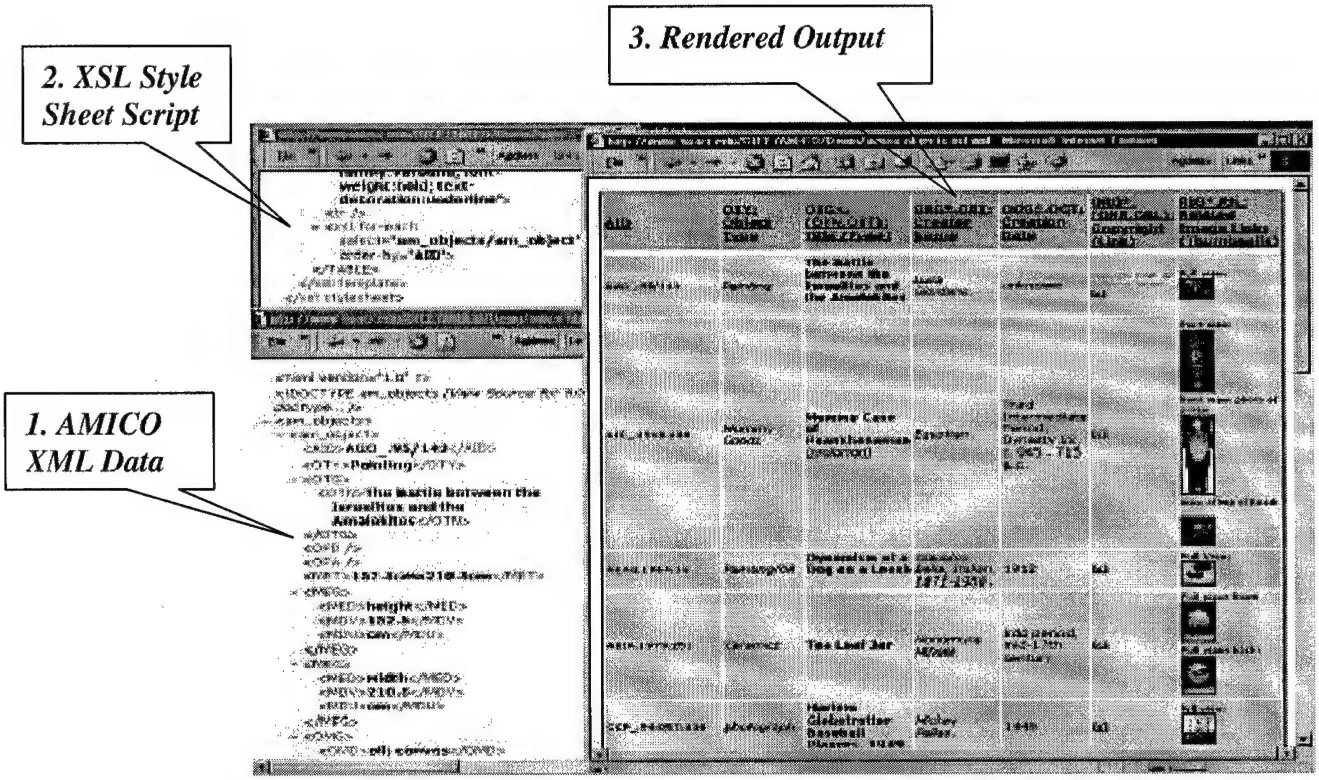
The query “Find the title, type, and image identifier of paintings” can be expressed in XMAS as follows:

```
CONSTRUCT <my_object>
            <my_title> $Title </my_title>
            <my_type> $Type </my_type>
            <my_img> $Img </my_img>
</my_object>
WHERE
<am_objects>
<am_object>
            <OTY> $Type </OTY>
            <OTG>
            <OTN> $Title </OTN>
            </OTG>
            <RIG>
            <RIL> $Img </RIL>
            </RIG>
</am_object>
</am_objects>
IN “http://www.npaci.edu/DICE/AMICO/Demo/amico-objects.xml”
AND substr(“Painting”, $Type)
```

This query is applied against the XML DTD tagged data.

Demo 4. Presenting XML

One of the main advantages of using XML is the separation of content (data has “semantic” tags) and presentation. A standard way to handle the latter are style sheets, which prescribe how the different elements are to be displayed. For the demo we used XSL style sheets. Fig. 12-2 shows the input and output of the XML presentation.



```
        weight:bold; text-
        decoration:underline">
      </th>
    - <xsl:for-each
        select="am_objects/am_object"
        order-by="AID">
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<!DOCTYPE am_objects (View Source for full
doctype...)>
- <am_objects>
  - <am_object>
      <AID>AGO_.85/143</AID>
    <OTY>Painting</OTY>
  - <OTS>
      <OTN>The Battle between the
        Israelites and the
        Amalekites</OTN>
    </OTS>
    <OPD />
    <OPA />
    <MET>157.5cmx210.5cm</MET>
  - <MEG>
      <MED>height</MED>
      <MDV>157.5</MDV>
      <MDU>cm</MDU>
    </MEG>
  - <MEG>
      <MED>width</MED>
      <MDV>210.5</MDV>
      <MDU>cm</MDU>
    </MEG>
  - <OMG>
      <OMD>oil; canvas</OMD>
```

AID	OTY: Object Type	OTGS (OTN, OTT): Title (Type)	OBGS.OBI: Creator Name	OOGS.OCT: Creation Date	ORGP (ORR,ORL): Copyright (Link)	RSG*.RS: Related Image Links (Thumbnails)
[illegible]	Painting	The Battle between the Israelites and the Amalekites	Luca Giordano	unknown	(c)	Full view
AIC_[illegible]	Mummy Goods	Mummy Case of Paankhenamun (Exterior)	Egyptian	Third Intermediate Period, Dynasty 22, c. 945 - 715 B.C.	(c)	[illegible]
[illegible]	Painting/Oil	Dynamism of a Dog on a Leash	Giacomo Balla, Italian, 1871-1958	1912	(c)	Full View
[illegible]	Ceramic	Tea Leaf Jar	Anonymous Artist	Edo period, [illegible] century	(c)	[illegible]
CCF_[illegible]	photograph	Harlem Globetrotter Baseball Players, [illegible]	[illegible]	1948	(c)	Full view

Figure 12.2. Application of XML Style sheets

13. Collection support - JITC Collection

COLLECTION DESCRIPTION:

The collection contains a set of miscellaneous office automation products in proprietary file formats. Files include word processing documents (e.g. Microsoft Word files), images (e.g. gif, jpg), and other binary files such as NITF (National Imagery Transfer Format) satellite imagery related data. This collection is intended to be an office automation suite of files that are used by the Defense Department to test high-volume servers. The collection features a mixture of proprietary formats.

OBJECTIVES:

Demonstrate *preservation strategy* for a heterogeneous collection of binary files.

RELATED COLLECTIONS:

Relates to the ***E-mail*** collection where attachments can be binary objects.

APPROACH TAKEN:

A collection of binary objects is archived using the DB2/HPSS system, where IBM's DB2 database has been integrated with the HPSS archival storage system. The meta-data is stored in columns of relational tables, while the binary objects are stored in columns of the same table. Using the functionality provided by DB2/HPSS, data in the latter columns are actually stored in HPSS as files.

IMPORTANT FINDINGS:

Important findings are the ability to index the archived collection on various meta-data attributes including, file type, file size, and file name.

13.1 Information model

The structure of this collection further detailed in Appendix I, facilitates the organization, archiving, indexing and access of file collections, based on file extension.

All the files in the JITC collection are in proprietary formats and are treated as binary objects. The meta-data associated with each of these binary objects includes filename, file size, and subject. If the filename contains a file extension, then that information is extracted and stored as additional meta-data. The meta-data values can be used to classify objects. Each object can be classified based on its file type attribute as well as on its file size attribute. These are orthogonal classifications. For example, since files with extension *ntf* range in size from 2Kbytes to 297Mbytes, they cover all three of the size grouping used to categorize the collection (0-100kB, 100kB-5 MB, 5MB-500MB). Similarly, of the 268 files with extension *doc,* all have file size less than 56Kbytes except one that has a size of 143Kbytes. As a result, the *doc* sub-collection covers two file size groupings (0-100K and 100K-5M).

13.2 Ingestion process

Figure 13-1 shows the procedure used for archiving this collection using the DB2/HPSS system where IBM's DB2 database has been integrated with the HPSS archival storage system. As shown in the figure, a table is defined for each storage classification. Files in the original collection are encapsulated as archival objects, which contain the corresponding meta-data, and stored in the appropriate table. While there are three separate tables—one for each file size grouping—all tables use the same *tablespace container* file in the archival storage system to store the archived object.

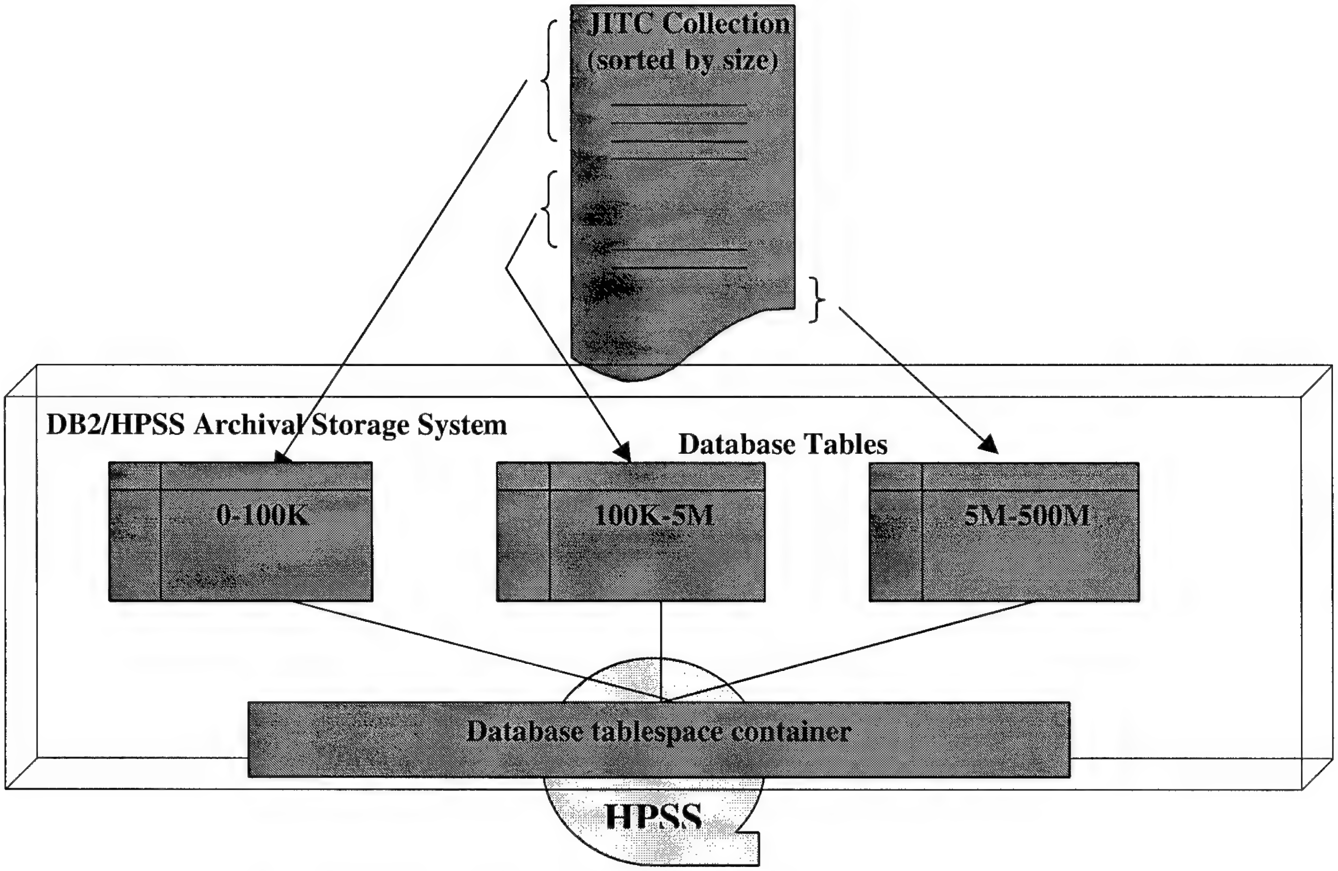


Figure 13-1. Storage strategy for the JITC collection

Performance

Information on the time taken to ingest the JITC collection into the SDSC archives is summarized below. The total time for ingesting the 680 files (388MB) was 17 minutes. Most of this time was for ingesting the single 297MB file in the collection. As shown in the table below, the overall ingestion rate for this JITC sample is 380Kbytes/sec and 40 files/minute. The I/O rate was limited by use of an Ethernet to access the HPSS storage system.

The above numbers must be interpreted carefully since (i) they represent only a single data point, (ii) the total number and size of files in this collection is quite small to be able to draw general inferences about performance, (iii) the ingestion experiment was run on distributed systems at SDSC that were using their "standard" configuration, i.e. none of the underlying systems including the operating system, network system, database system, and archival storage system were optimized for providing the best performance for this particular application.

# of files	Size in MB	Ingestion			
		Time	Byte rate	File rates	
				Files/min	10,000 files
680	388	17 mins.	380KB/sec	40 files/min	10,000 files in 10 hours

Table 13-1. Ingestion times for JITC collection

TASK OBJECTIVE

It was assumed that the heterogeneous collection of documents would be acquired from an existing government web site. However, due to various issues involved in acquiring data out of an active web site, and also given the fact that NARA was able to provide us this benchmark collection, we used this particular collection rather than acquiring one from a web site. This collection has been archived using the DB2/HPSS integrated archival system with the available meta-data as well as the data objects stored together as a single archival object. As mentioned earlier, it is possible to index the archived collection on various meta-data attributes including, file type, file size, and file name.

14. Remaining Technical Issues

The four major components of the persistent archive process are support for ingestion, archival storage, information discovery, and presentation of the collection. The first two components focus on the ingestion of data into collections. The last two focus on access to the resulting collections. The technology to support each of these processes is still rapidly evolving. Hence consensus on standards has not been reached for many of the infrastructure components. At the same time, many of the components are active areas of research. To reach consensus on a feasible collection-based persistent archive, continued research and development is needed. Examples of the many related issues are listed below:

Ingestion

- Creation of a standard digital representation of the original (or raw) data. What unique tags should be used to define digital objects within the original raw data?
- Techniques for automating the decomposition of a data collection into individual digital objects. How can digital objects be defined when they must be extracted from proprietary formats?
- Automation of the mining of attributes used to describe each data object. Can a generic technique be developed that works for a class of data such as E-mail, or word processing documents?
- Standard information model for characterization of the data collection organization. This will require defining standard semantics as well as a standard for describing the collection structure.
- Representation of unique procedures associated with each collection, including software access tools and ingestion update tools. Can these tools be made interoperable across multiple collections, or will unique tools be required for each collection?
- Standardization of the mark-up language used to annotate the digital objects with their associated meta-data. Extensions are being proposed to XML to associate semantics with the tags, and define required structures within the DTD.
- Support for security within the ingestion process. What risks are incurred by use of common infrastructure for ingesting data at different security levels?
- Validation of ingestion process. Policies for validating the correctness of an infrastructure independent representation of a digital object are needed. The XML approach does not capture white space.
- Workflow management policies are needed as a component of the ingestion process, to ensure that all validation steps are completed. Can validation be done after the fact through analysis of the collection, or should the validation be confirmed as the digital objects are created?
- Evolution of information models. There is a need for finding aids that are robust under evolution, and capable of locating all data collections stored within the persistent archive.
- Collection access. Access mechanisms are needed that are capable of handling changes to collections, such as construction of new indexes of collections, updating of

collections by addition of objects, updating of collections by addition of new attributes, and updating through evolution of the DTD.

- Performance optimization for incremental updates of collections, and incremental updates of DTDs.
- Administrative tools for managing collections, including compaction of collections, updates to collections, and restructuring of collections.
- Derivation of DTDs to describe complex, semi-structured and unstructured collections.
- Support for heterogeneous collections, especially multi-media, graphical and web-based collections

Archival storage

- Standardization of the archive format for storing a digital object based upon OAIS
- Standardization of container formats for aggregating digital objects
- Choice of digital objects to aggregate within containers for retrieval optimization from the archive
- Standardization for registration of the collection within a finding aid to guarantee the ability to retrieve the data collection from the archive
- Versioning of the information model to track changes
- Support for migration of data between time dependent security levels

Information discovery

- Development of generic software that is able to parse DTDs and generate appropriate commands for creating a new database.
- Support for dynamic reconstruction of the data collection through use of XML-based database technology. Additional attributes may need to be defined to manage evolution of a hierarchical structure.
- Support for dynamic generation of a user interface to support queries against data in XML databases.
- Dynamic generation of the query language required to access XML databases.
- Support for creation of queries against collections that have evolved
- Access control mechanisms and standards for discovering classified information.

Presentation

- Standardization of the mark-up language used to define the presentation layout (XSL style sheets). An example is optimization of the layout of the display for user efficiency and ease of use
- Support for retrieval and presentation of meta-data used to characterize information about the object or the associated data collection. Can standard DTDs be used to organize meta-data for presentation?
- Dynamic creation of the presentation interface for each digital object. Presentation cannot be a function of only the collection. For heterogeneous collections, the style of presentation must be defined for each type of object within the collection.

14.1 Research Opportunities

Important research areas that we suggest pursuing include:

- **Security**: So far we have dealt with unencumbered data ingestion. What happens when particular data elements need to reside at certain locations or when notions of data access control come into play?
- **Federation**: Can a persistent archive be distributed? DTD interoperability and integration are associated topics.
- **Workflow**: The validation process requires the guaranteed execution of analysis routines. Workflow management tools are needed to ensure that no processing steps are missed.
- **Complex collections**: What is the correct information model when dealing with multimedia data and GIS collections?

14.2 Research and Development Tasks

The tasks described within this paper represent initial efforts on a project funded by the National Archives and Records Administration as an extension to the DARPA/USPTO Distributed Object Computation Testbed (DOCT) grant F19628-96-C-0020. The projects are described by a Research and Development Plan and Schedule, and are detailed in Table 14-1. Progress has been made on all tasks, but as described above, there are many research issues that still need to be explored.

Task	Sections
1.A – Ingestion of million record E-mail collection	5
1.B – Ingestion of word processing collection	13
1.C – Ingestion of heterogeneous collections	6,7,8,9,10,12,13
1.D – Ingestion of Patent collection	11
II.A – Persistent storage – performance modeling	3.1, 3.2
II.B – Persistent storage – technology evolution	3.1
III.A – Archiving meta-data – schema evolution	3.1
III.B – Archiving meta-data – ontology evolution	3.1

Table 14-1. Research Projects

The ingestion tasks were augmented to include additional collections. The goal was to demonstrate applicability of the technology across a wide selection of collections, which resulted in the examination of a total of nine collections. The research on persistent storage was demonstrated primarily through the ingestion, storage, and retrieval of a million-record E-mail collection. The research on archiving meta-data was implemented in the MCAT system meta-data management software. This required identifying the meta-schema attributes that are needed to manage a schema, and implementing software that can use the meta-schema attributes to instantiate a collection. The ontology evolution task has multiple components, related to provision of multiple views into a data collection, and support for evolution of the information model used to describe collections. Both areas continue to be important research areas.

14.3 Summary

Multiple communities across academia, the federal government, and standards groups are exploring strategies for managing very large archives. The persistent archive community needs to maintain interactions with these communities to track development of new strategies for data management and storage. The technology proposed by SDSC for implementing persistent archives builds upon interactions with many of these groups. Explicit interactions include publications with Federal planning groups [21], the Computational Grid [22], the digital library community [23], and individual federal agencies [24].

The proposed persistent archive infrastructure combines elements from supercomputer centers, digital libraries, and distributed computing environments. The synergy that is achieved can be traced to identification of the unique capabilities that each environment provides, and the construction of interoperability mechanisms for integrating the environments. The result is a system that allows the upgrade of the individual components, with the ability to scale the capabilities of the system by adding resources. By differentiating between the storage of the information content and the storage of the bits that comprise the digital objects, it is possible to create an infrastructure independent representation for data organized by collections. Collection-based persistent archives are now feasible that can manage the massive amounts of information that confront government agencies.

Acknowledgements:

The data management technology has been developed through multiple federally sponsored projects, including the DARPA project F19628-95-C-0194 “Massive Data Analysis Systems,” the DARPA/USPTO project F19628-96-C-0020 “Distributed Object Computation Testbed,” the Data Intensive Computing thrust area of the NSF project ASC 96-19020 “National Partnership for Advanced Computational Infrastructure,” the NASA Information Power Grid project, and the DOE ASCI/ASAP project “Data Visualization Corridor.” Additional projects related to the NSF Digital Library Initiative Phase II, the California Digital Library at the University of California, and NSF Digital Government are just starting, and will also support the development of information management technology. This work was supported by a NARA extension to the DARPA/USPTO Distributed Object Computation Testbed, project F19628-96-C-0020. Michael Wan led the research and development of the Storage Resource Broker. Wayne Schroeder led the research and development of the security infrastructure. The XMAS research and development was led by Yannis Papakonstantinou of the University of California, San Diego.

References

1. Rajasekar, A., Marciano, R., Moore, R., "Collection Based Persistent Archives," Proceedings of the 16th IEEE Symposium on Mass Storage Systems, March 1999.
2. Extensible Markup Language, http://www.w3.org/XML
3. The High Performance Storage System (HPSS), http://www.sdsc.edu/HPSS/.
4. Transaction Processing Council, http://www.tpc.org/results/tpc_d.results.page.html
5. Baru C., Moore, R., Rajasekar, A., and Wan, M., "The SDSC Storage Resource Broker," *Proceedings of CASCON'98 Conference*, Nov. 30–Dec. 3, 1998, Toronto, Canada.
6. Baru C., Frost, R., Marciano, R., Moore, R., Rajasekar, A., and Wan, M., "Meta-data to support information based computing environments," *Proceedings of the IEEE Conference on Meta-data*, Silver Spring, MD, Sept. 1997.
7. MCAT – A Meta Information Catalog (V1.1), Technical report: http://www.npaci.edu/DICE/SRB/mcat.html
8. Data Definition Language standardized syntax, ANSI X3.135-1992 (R1998)
9. Data-Intensive Computing Environment reports, URL http://www.npaci.edu/DICE/.
10. The DB2/HPSS Integration project, http://www.sdsc.edu/MDAS.
11. IEEE Storage System Standards Working Group (SSSWG) Project 1244, "Reference Model for Open Storage Systems Interconnection, Mass Storage Reference Model Version 5," Sept. 1994.
12. Moore, R., Lopez, J., Lofton, C., Schroeder, W., Kremenek, G., Gleicher, M., "Configuring and Tuning Archival Storage Systems," Proceedings of the 16th IEEE Symposium on Mass Storage Systems, March 1999.
13. Schroeder W., "The SDSC Encryption / Authentication (SEA) System," Distributed Object Computation Testbed (DOCT) project white paper, http://www.sdsc.edu/~schroede/sea.html.
14. Schroeder W., "The SDSC Encryption and Authentication (SEA) System," Special Issue of Concurrency: Practice and Experience—Aspects of Seamless Computing, John Wiley & Sons Ltd., 1999.
15. Baru C., and Rajasekar, A., "A Hierarchical Access Control Scheme for Digital Libraries," *Proceedings of the 3rd ACM Conference on Digital Libraries*, Pittsburgh, PA, June 23-25, 1998.
16. The Dublin Core, http://www.ukoln.ac.uk/meta-data/resources/dc.html.
17. The Warwick Framework, Carl Lagoze, *D-Lib Magazine*, July/August 1996, http://www.dlib.org/dlib/july96/lagoze/o7lagoze.html
18. Tomer, C., "Information Technology Standards for Libraries," *Journal of the American Society for Information Science*. 43: 566-570, 1992.
19. Light-Weight Directory Access Protocol (LDAP) implementation by Netscape, http://www.netscape.com/comprod/server_central/-product/directory/index.html
20. Schroeder, W., Marciano, R., "Workload characterization of HPSS," Proceedings of the 16th IEEE Symposium on Mass Storage Systems, March 1999.
21. Moore, R., "Enabling petabyte computing, The Unpredictable Certainty, Information Infrastructure through 2000," National Academy Press, 1997.
22. Foster, I., Kesselman, C., "The Grid: Blueprint for a New Computing Infrastructure," Chapter 5, "Data-intensive Computing," Morgan Kaufmann, San Francisco, 1999.

23. Baru C. "Archiving Meta-data," 2nd European Conference on Research and Advanced Technology for Digital Libraries (poster), Sept.19-23, 1998, Crete, Greece.

24. Baru, C., *et al.*, "A data handling architecture for a prototype federal application," *Proceedings of the IEEE Conference on Mass Storage Systems*, College Park, MD, March 1998.

APPENDIX A: E-mail Postings

PHYSICAL SOURCE:

A live newsgroup feed from SDSC's Newsgroup Server was tapped to generate the 1 million E-mail message collection. Rather than accessing data from a newsgroup archive, this approach was preferred in order to learn the issues involved with the archiving of "live" data feeds.

Four principal newsgroup categories were selected: *comp, humanities, sci, soc*, as they contain, respectively, computer messages, humanities messages, general science messages, and social science messages. Each category was monitored for multiple newsgroups, with messages archived from 1,170 separate newsgroups.

Comp	733
Humanities	7
Sci	184
Soc	246
	1,170

Table A-1: Total newsgroup count monitored

The collecting of *1 million* Newsgroup messages into a file called *collection.raw* was completed on January 16, 1999, in *3 ½ weeks*. Typically *3 days* worth of accumulated live feed for these 4 categories of newsgroups (1,170 individual groups) can be assembled in *1 ½ hours.*

Incidentally, this represents *less than 4% of all the newsgroups* received at SDSC and *less than 9% of all the messages* stored at a given point in time (total live feed stats: *1 million records on disk* and *30,754 newsgroups* subscribed to). Table A-2 lists the number of messages received for each newsgroup category, the total size, the percentage of the collection that came from the category, the number of messages received per week, and the average message size. The total collection size was 2.5 GB of raw data.

	Num.	***Total Size***	***Perc.***	***Groups***	***Examples***	***Num/ Week***	***Msg. Size***
Comp	533,956	1.13 GB	54.0%	733	comp.ai.fuzzy comp.benchmark comp.databases.oracle.tools	144,000	2.2 KB
Humanities	4,510	13.4 MB	0.5%	7	humanities.languages.sanskrit humanities.lit.authors	1,200	3.0 KB
Sci	113,428	285.0 MB	11.5%	184	sci.space.shuttle sci.physics.plasma sci.fractals	31,000	2.3 KB
Soc	348,106	1.1 GB	34.0%	246	soc.genealogy.surnames.usa soc.history.war.vietnam	94,000	3.3 KB
	1,000,000	**2.52 GB**	**100%**	**1,170**		**281,000**	**2.7**

							KB

Table A-2. Newsgroup E-mail statistics

It appears that “humanities” and “social science” messages are on the average a little longer (3.1 & 3.2 KB), which could indicate that scientists and engineers are a little more terse! ☺

Individual message objects follow the Network Working Group *RFC-1036* (http://http.bsd.uchicago.edu/~twpierce/news/rfc1036.html) standard, a standard for Interchange of Usenet Messages. A *standard USENET message* consists of several *header lines*, followed by a blank line, followed by the *body* of the message.

Certain headers are *required*, and certain headers are *optional*. Any *unrecognized headers* are allowed, and will be passed through unchanged. Each header line consists of a keyword, a colon, a blank, and some additional information. The Internet convention of continuation header lines (beginning with a blank or tab) is allowed.

Table A-3 gives a field count tally over the 1 million record collection we assembled. The results are in accordance with RFC1036, where there are 6 required fields, 13 optional fields, and a variable number of unrecognized headers (we call this last category: *other*). Note that 1,347 user-defined headers were present in the collection. All Usenet messages had all of the required fields.

	Required	***Optional***	***Other***
	From:	Followup-To:	X-spam-hater:
	Date:	Expires:	X-WebTV-Signature
	Newsgroups:	Reply-To:	X-Christmas:
	Subject:	Sender:	X-Coffee
	Message-ID:	References:	Return-Path:
	Path:	Control:	Status:
		Distribution:	Originator:
		Keywords:	Abuse-Reports-To:
		Summary:	X-truth:
		Approved:	Resent-To:
		Lines:	X-A-Notice:
		Xref:	X-No-Hope:
		Organization:	X-001:
			Battlestar-Galactica-Date:
			Etc…
Counts	**6**	**13**	***Variable Count*: 1,347**

Table A-3. Summary of fields in the Usenet E-mail messages

APPENDIX B: TIGER/Line'92 additional information

PHYSICAL SOURCE:

The data is packaged as 44 CDs with up to 600 MB / CD (Total size: 26 GB). The Tiger collection is by far the most intricate in terms of data organization and associated meta-data.

The rest of this section is devoted to providing an in-depth view of the TIGER/Line data model.

Topological Model

The Census TIGER database represents a seamless national file with no overlaps or gaps between parts. However, each county-based TIGER/Line file is designed to stand alone as an independent data set or the files can be combined to cover the whole Nation and its territories. The data collection represent the topological information associated with the Census data, and is organized as points (nodes) / lines (chains of points) / closed polygons.

The 1992 TIGER/Line files consist of line segments representing physical features and governmental and statistical boundaries. The files contain information distributed over a series of record types for the spatial objects of a county. There are *14 record types*, including the basic data record, the shape coordinate points, and geographic codes that can be used with appropriate software to prepare maps. Other geographic information contained in the files includes attributes such as feature identifiers/census feature class codes (CFCC) used to differentiate feature types, address ranges and ZIP Codes, codes for legal and statistical entities, latitude/longitude coordinates of linear and point features, landmark point features, area landmarks, key geographic features, and area boundaries. The 1992 TIGER/Line *data dictionary* contains a complete list of all the fields in the 14 record types.

CD-ROM Structure

A typical CD-ROM containing information for sake of example, for 2 states, will have the following directory and file structure, shown in Table B-1.

```
/Tools
/Software
/Document
Readme
/STATE1#
          /COUNTY1#
```

```
            . . .
            /COUNTYn#
/STATE2#
            /COUNTY1#
            . . .
            /COUNTYn#
```

Table B-1. Example of CD-ROM format for Tiger/Line data

There are up to 14 separate files for each county directory:
/ST/CTY
where: ST = [FIPS State Code]
CTY = [FIPS County Code]

The following CD-ROM snapshot shows portions of **Texas** (FIPS code: **48**) and **Oklahoma** (FIPS code: **40**):

```
/Tools
   Filename                  Description

      Readme
      RT?.DBF                dBASE file structures for 14 TIGER/Line record types.
                             The TIGER/Line files are found in the county
                             directory within the state directory, where:
                                    ?=(1..8,a,f,g,I,p,r).
                                    1: single record for each unique complete chain
                                    2: additional series of lat/long values
                                    3: additional 80 & 90 entity codes
                                    4: index to alternate feature id's
                                    5: feature identifier list
                                    6: additional address range and ZIP code data
                                    7: landmark features
                                    8: polygons linked to area landmarks
                                    a: additional polygon geographic entity codes
                                    f: corrected geog. Entity codes for the 90 Census
                                    g: 1992 geographic codes & entity changes
                                    i: the link between complete chains and polygons
                                    p: polygon internal point
                                    r: record number range
      SCHLDNAM.DBF           dBASE file structure for the School District
      SCHOLDxx.NAM           Names file, found in the state directory (with the
                             State's FIPS code as the directory name.)
      TGR92NAM.DBF           dBASE file structure for the geographic reference
      TGR92Sxx.NAM           file found in the state directory.
      TG92NRTS.DBF           dBASE file with the record type codes and
                             descriptions for file TGR92NAM.DBF.
/40

      /001
```

```
                Tgr40001
                Tgr40001.f52
                ...
                Tgr40001.f58
                Tgr40001.f5a
                Tgr40001.f5g
                Tgr40001.f5I
                Tgr40001.f5p
                Tgr40001.f5r

        /003
        ...
        /151
        /153
        Okcensus.dbf
        Okplaces.dbf
        Schold40.nam
        Tgr92s40.nam
/48

        /491

                Tgr48491
                Tgr48491.f52
                ...
                Tgr48491.f58
                Tgr48491.f5a
                Tgr48491.f5f
                Tgr48491.f5g
                Tgr48491.f5I
                Tgr48491.f5p
                Tgr48491.f5r

        /493
        ...
        /507
        Schold48.nam
        Tgr92s48.nam
        Txcensus.dbf
        Txplaces.dbf
/Document

        /Tiger

                /Ascii
                /Graphics
                /Mac
                /Wp

        Landview.doc
        Schldnam.doc
        Tgr92nam.doc
        Usrguide.doc
Readme
```

```
/Software

      Landview.exe              (& related files: install.exe, etc.)
```

Table B-2. County data format for Tiger/Line

For 1992, there are 3,428 county data sets. These constitute the fundamental digital objects stored within the collection. Each digital object is well-defined in the collection, with the data for each county is stored in 14 files.

Files contain three major types of data:

Line features:

- roads
- railroads
- hydrography
- misc. transportation features & selected power lines & pipe lines
- boundaries

Landmark

- point: schools, churches
- area: parks, cemeteries

Polygon

- geographic entity codes locations of area landmarks

The following 20 pages describe the data in greater detail:

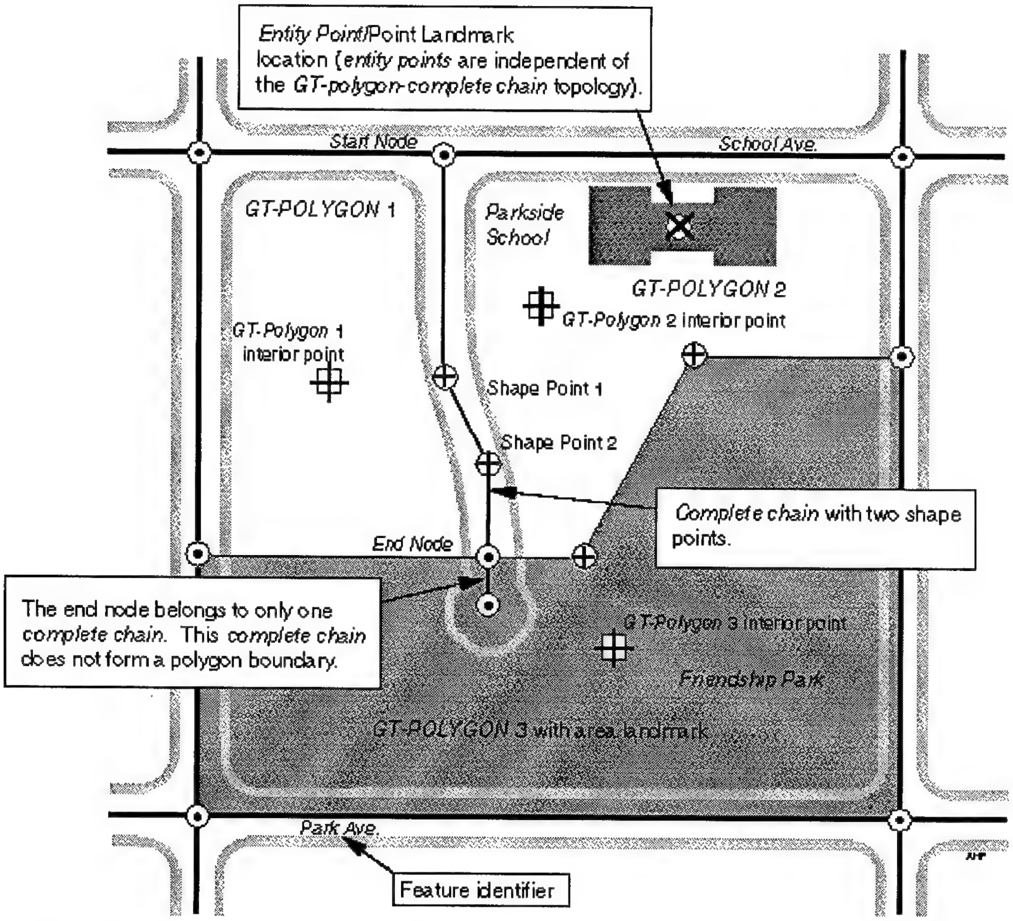
- **6 pages / figures** give a pictorial view of the structure of Tiger files:
    - *Figure 1-1:* gives the Basic TIGER/Line File Topology
    - *Figure 1-2:* gives the TIGER/Line File Record Linkages
    - *Figure 3-1:* shows the TIGER/Line Address Range Basics
    - *Figure 4-1:* shows the Hierarchical Relationships of Geographic Entities in the TIGER/Line Files
    - *Figure 4-2:* shows Geographic Relationships for Small Area Statistical Entities
    - *Additional Figure:* shows Geographic Relationships for Legal & Statistical Entities.

- 14 pages / sections provide a detailed *data dictionary* that describes each of the 14 associated county file components: 1,2,3,4,5,6,7,8,a,f,g,I,p,r.

Figure 1-1

Basic TIGER/Line™ File Topology

The illustration below shows a generalized block that consists of 3 *GT-polygons* (*GT* stands for geometry and topology). The block contains a point landmark (Parkside School) and an area landmark (Friendship Park) that is coextensive with *GT-polygon* 3.


Entity Point/Point Landmark location (*entity points* are independent of the *GT-polygon-complete chain* topology).
Start Node
School Ave.
GT-POLYGON 1
Parkside School
GT-POLYGON 2
GT-Polygon 2 interior point
GT-Polygon 1 interior point
Shape Point 1
Shape Point 2
Complete chain with two shape points.
End Node
The end node belongs to only one *complete chain*. This *complete chain* does not form a polygon boundary.
GT-Polygon 3 interior point
Friendship Park
GT-POLYGON 3 with area landmark
Park Ave.
Feature identifier

▨ Actual Street Curb Location

⊙ TIGER/Line™ *Node*: A zero-dimensional object that incorporates topology and geometry. Each marks the intersection or end point of a *complete chain*.

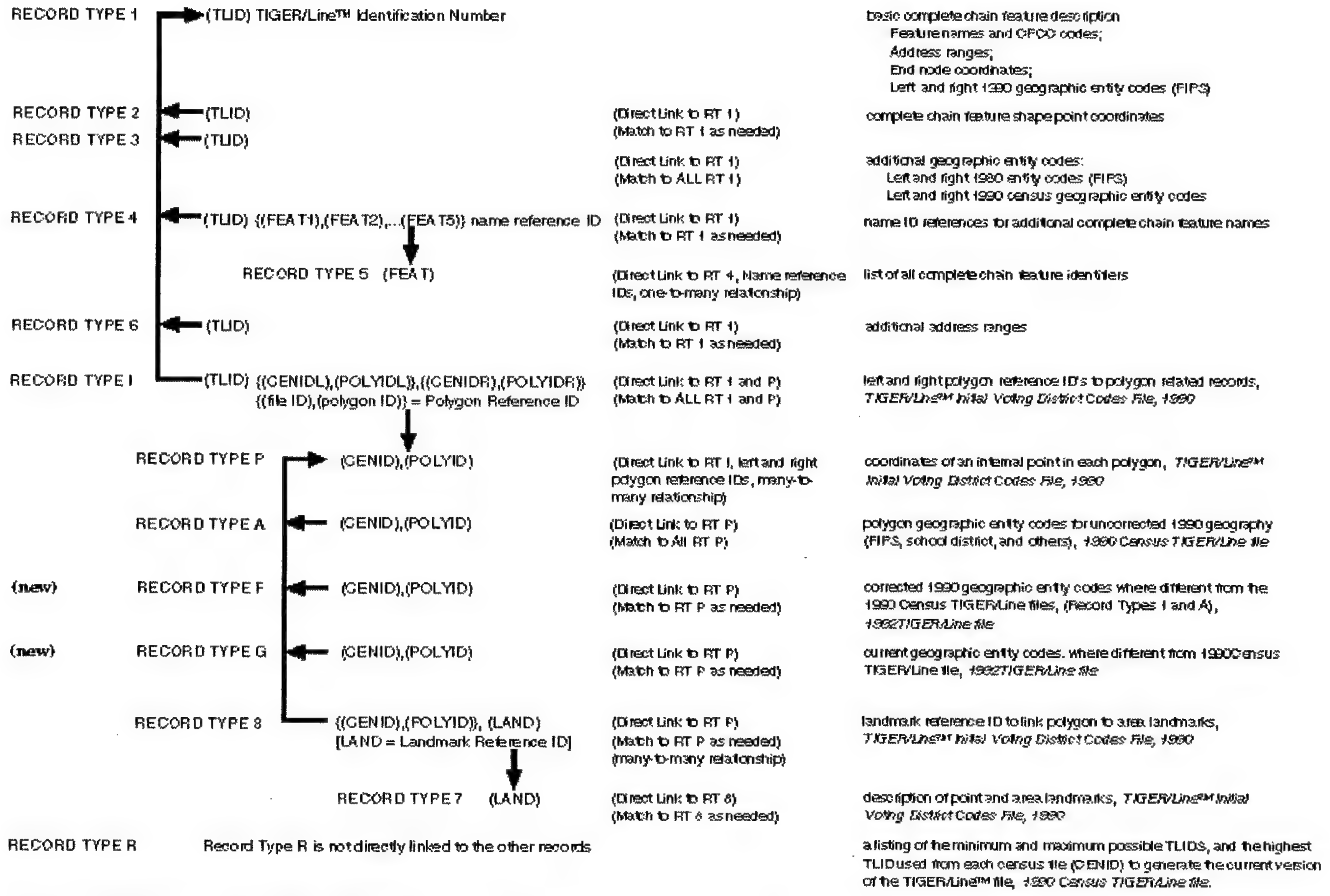
⊕ TIGER/Line™ Shape Point: A zero-dimensional object that defines the curvature of a *complete chain*, but is not required to describe the topology of the *complete chain* (unlike the intersections and end points).

☒ TIGER/Line™ Point Landmark: An *entity point* that identifies the location of a point landmark, in this case the Parkside School.

⌖ TIGER/Line™ Polygon Interior Points: An *area point* associated with a polygon in the TIGER/Line™ file.

⊙—⊙ *Complete Chain*: A one-dimensional object having topological and geometric characteristics.

Figure 1-2
TIGER/Line™ File Record Linkages



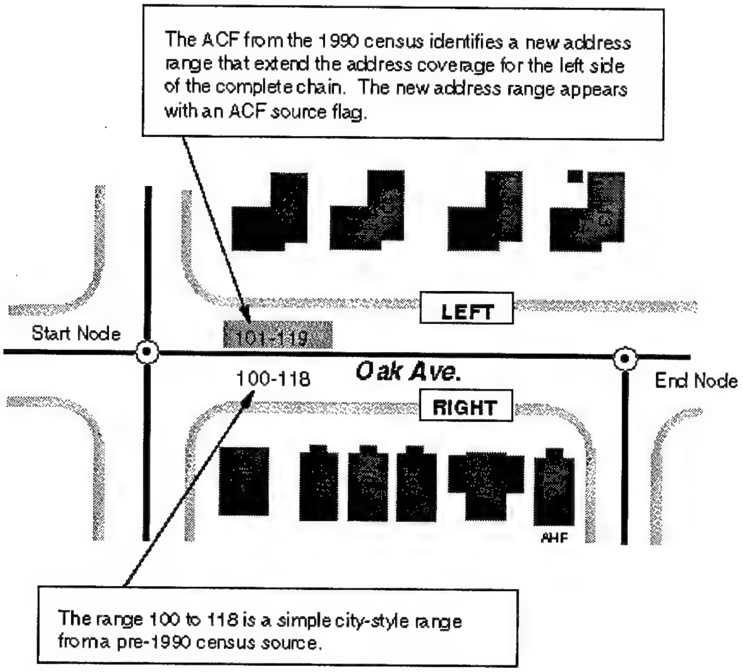
	RECORD TYPE 1	(TLID) TIGER/Line™ Identification Number		basic complete chain feature description
				Feature names and CFCC codes;
				Address ranges;
				End node coordinates;
				Left and right 1990 geographic entity codes (FIPS)

	RECORD TYPE 2	(TLID)	(Direct Link to RT 1) (Match to RT 1 as needed)	complete chain feature shape point coordinates
	RECORD TYPE 3	(TLID)	(Direct Link to RT 1) (Match to ALL RT 1)	additional geographic entity codes: Left and right 1980 entity codes (FIPS) Left and right 1990 census geographic entity codes
	RECORD TYPE 4	(TLID) {(FEAT1),(FEAT2),...(FEAT5)} name reference ID	(Direct Link to RT 1) (Match to RT 1 as needed)	name ID references for additional complete chain feature names
	RECORD TYPE 5	(FEAT)	(Direct Link to RT 4, Name reference IDs, one-to-many relationship)	list of all complete chain feature identifiers
	RECORD TYPE 6	(TLID)	(Direct Link to RT 1) (Match to RT 1 as needed)	additional address ranges
	RECORD TYPE I	(TLID) {(CENIDL),(POLYIDL)},{(CENIDR),(POLYIDR)} {(file ID),(polygon ID)} = Polygon Reference ID	(Direct Link to RT 1 and P) (Match to ALL RT 1 and P)	left and right polygon reference ID's to polygon related records, *TIGER/Line™ Initial Voting District Codes File, 1990*
	RECORD TYPE P	(CENID),(POLYID)	(Direct Link to RT I, left and right polygon reference IDs, many-to-many relationship)	coordinates of an internal point in each polygon, *TIGER/Line™ Initial Voting District Codes File, 1990*
	RECORD TYPE A	(CENID),(POLYID)	(Direct Link to RT P) (Match to All RT P)	polygon geographic entity codes for uncorrected 1990 geography (FIPS, school district, and others), *1990 Census TIGER/Line file*
(new)	RECORD TYPE F	(CENID),(POLYID)	(Direct Link to RT P) (Match to RT P as needed)	corrected 1990 geographic entity codes where different from the 1990 Census TIGER/Line files, (Record Types 1 and A), *1992TIGER/Line file*
(new)	RECORD TYPE G	(CENID),(POLYID)	(Direct Link to RT P) (Match to RT P as needed)	current geographic entity codes. where different from 1990Census TIGER/Line file, *1992TIGER/Line file*
	RECORD TYPE 8	{(CENID),(POLYID)}, (LAND) [LAND = Landmark Reference ID]	(Direct Link to RT P) (Match to RT P as needed) (many-to-many relationship)	landmark reference ID to link polygon to area landmarks, *TIGER/Line™ Initial Voting District Codes File, 1990*
	RECORD TYPE 7	(LAND)	(Direct Link to RT 8) (Match to RT 8 as needed)	description of point and area landmarks, *TIGER/Line™ Initial Voting District Codes File, 1990*
	RECORD TYPE R	Record Type R is not directly linked to the other records		a listing of the minimum and maximum possible TLIDS, and the highest TLID used from each census file (CENID) to generate the current version of the TIGER/Line™ file, *1990 Census TIGER/Line file.*

Italics indicate the first version of the TIGER/Line™ file to contain the record type. Record Types 1 through 6 were the original record types from the Precensus version.

Figure 3-1
TIGER/Line™ Address Range Basics

The TIGER/Line™ files contain potential address ranges for city-style addresses. The complete chain below has two address ranges; the left side has an odd-number parity and the right side has a complimentary even-number parity.


The ACF from the 1990 census identifies a new address range that extend the address coverage for the left side of the complete chain. The new address range appears with an ACF source flag.

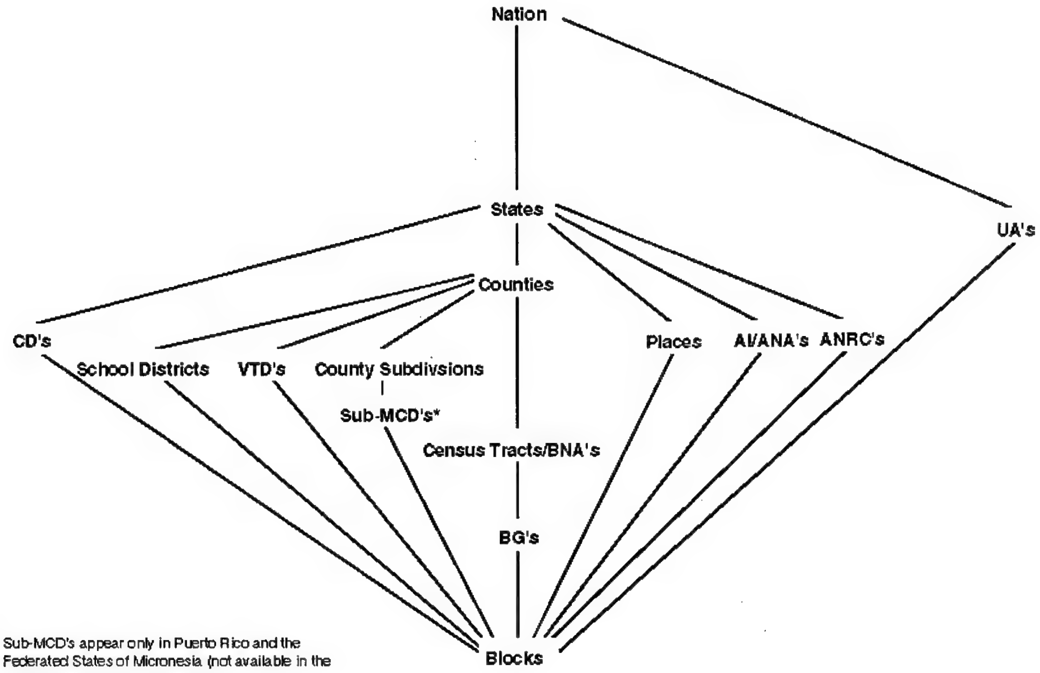
The range 100 to 118 is a simple city-style range froma pre-1990 census source.

Record Type 1 contains separate data fields for the start and end of each address range. Note that an impute flag with a value of "2" or "3" indicates that the address range is from the ACF source.

TIGER/Line™ Record Types				Address Range				Impute Flags			
				Left side		Right Side		Left side		Right Side	
				Start	End	Start	End	Start	End	Start	End
Record Type 1											
RT	TLID...	FENAME	FETYPE..	FRADDL	TOADDL	FRADDR	TOADDR	FRIADDFL	TOIADDFL	FRIADDFR	TOIADDFR
1...	0007654320...	Oak	Ave...	101	119	100	118	2	2	0	0

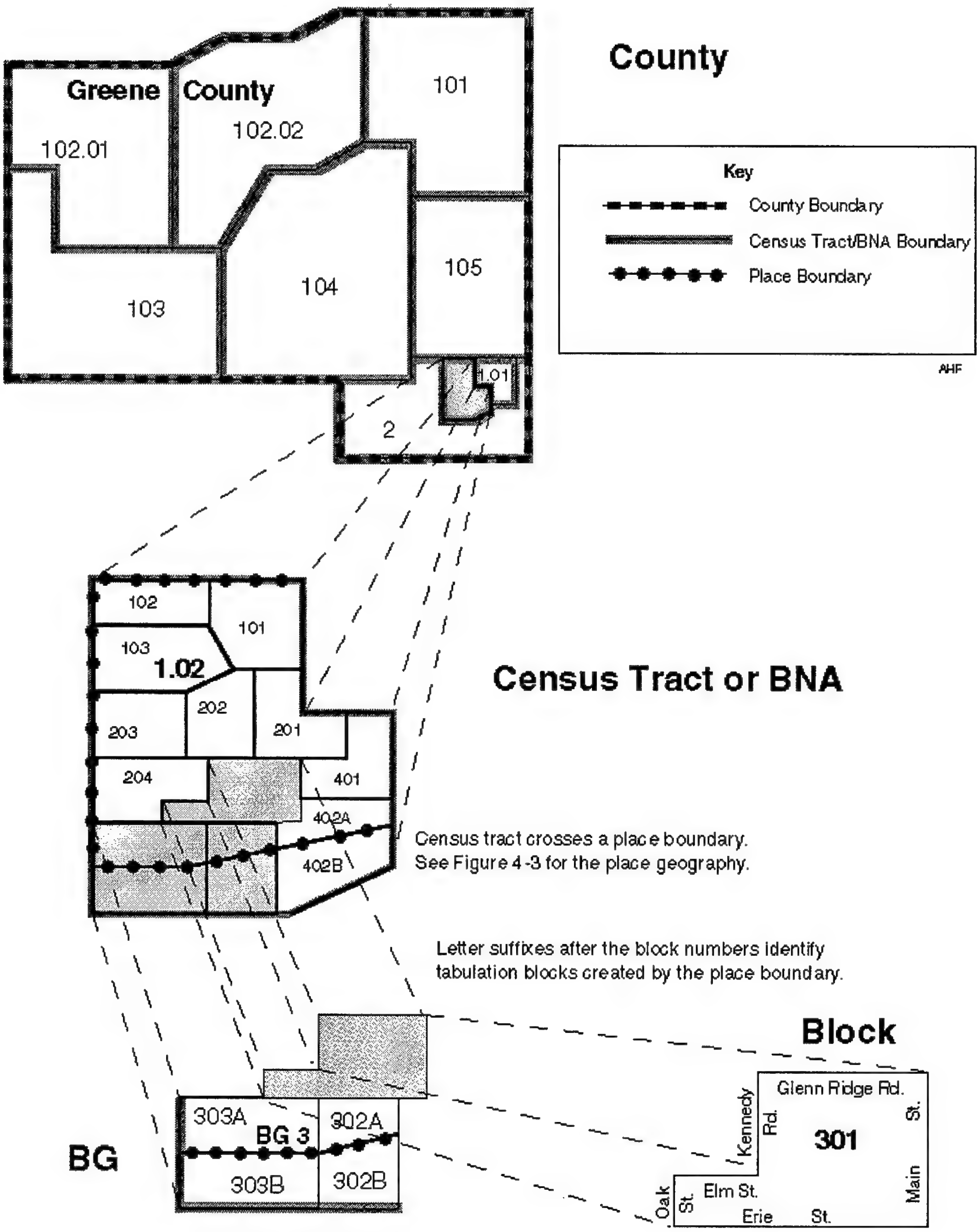
Figure 4-1

Hierarchical Relationships of Geographic Entities in the TIGER/Line Files



* Sub-MCD's appear only in Puerto Rico and the Federated States of Micronesia (not available in the TIGER/Line Files, 1992 Version).

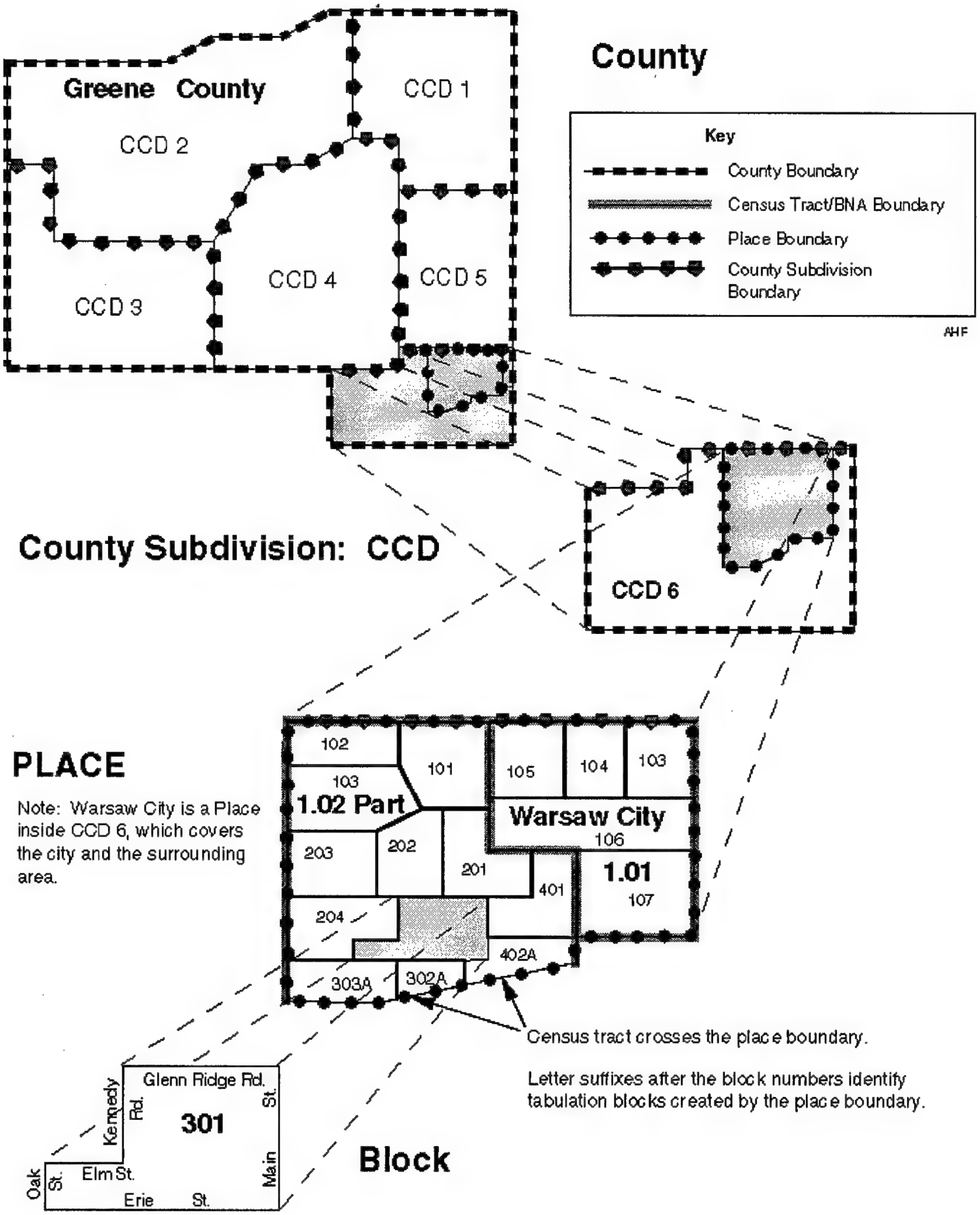
Figure 4-2
Geographic Relationships -- Small Area Statistical Entities



Census tract crosses a place boundary.
See Figure 4-3 for the place geography.

Letter suffixes after the block numbers identify tabulation blocks created by the place boundary.

Geographic Relationships -- Legal and Statistical Entities



Note: Warsaw City is a Place inside CCD 6, which covers the city and the surrounding area.

Census tract crosses the place boundary.

Letter suffixes after the block numbers identify tabulation blocks created by the place boundary.

The following pages detail the structure of each of the 14 files (1, .., 8, a, f, g, I, p, r).

1992 TIGER/Line Files
Data Dictionary

Record Type 1 -- Basic Data Record for Complete Chains

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	**L**	**A**	**1**	**1**	**1**	**N**	**Record Type (Record Type is "1")**
VERSION	**L**	**N**	**2**	**5**	**4**	**N**	**Version Number (Value "0005" identifies the 1992 TIGER/Line files)**
TLID	**R**	**N**	**6**	**15**	**10**	**N**	**TIGER/Line™ Record ID Number (Permanent complete chain identification number)**
1SIDE	**R**	**N**	**16**	**16**	**1**	**Y**	**Single-Side Segment Code (Value "1" signifies data only exists for one side of the complete chain)**
SOURCE	**L**	**A**	**17**	**17**	**1**	**Y**	**Source Code**
FEDIRP	**L**	**A**	**18**	**19**	**2**	**Y**	**Feature Direction, Prefix**
FENAME	**L**	**A**	**20**	**49**	**30**	**Y**	**Feature Name**
FETYPE	**L**	**A**	**50**	**53**	**4**	**Y**	**Feature Type**
FEDIRS	**L**	**A**	**54**	**55**	**2**	**Y**	**Feature Direction, Suffix**
CFCC	**L**	**A**	**56**	**58**	**3**	**Y**	**CFCC**
FRADDL	**R**	**A**	**59**	**69**	**11**	**Y**	**Start Address, Left Side**
TOADDL	**R**	**A**	**70**	**80**	**11**	**Y**	**End Address, Left Side**
FRADDR	**R**	**A**	**81**	**91**	**11**	**Y**	**Start Address, Right Side**
TOADDR	**R**	**A**	**92**	**102**	**11**	**Y**	**End Address, Right Side**
FRIADDL	**R**	**N**	**103**	**103**	**1**	**Y**	**Start Address, Impute Flag Left Side**
TOIADDL	**R**	**N**	**104**	**104**	**1**	**Y**	**End Address, Impute Flag Left Side**
FRIADDR	**R**	**N**	**105**	**105**	**1**	**Y**	**Start Address, Impute Flag Right Side**
TOIADDR	**R**	**N**	**106**	**106**	**1**	**Y**	**End Address, Impute Flag Right Side**
ZIPL	**L**	**N**	**107**	**111**	**5**	**Y**	**ZIP Code®, Left Side (A non-blank value appears only when left address range is present)**
ZIPR	**L**	**N**	**112**	**116**	**5**	**Y**	**ZIP Code®, Right Side (A non-blank value appears only when right address range present)**
FAIRL	**L**	**N**	**117**	**121**	**5**	**Y**	**AI/ANA FIPS PUB 55-3 Code, Left Side**
FAIRR	**L**	**N**	**122**	**126**	**5**	**Y**	**AI/ANA FIPS PUB 55-3 Code, Right Side**
ANRCL	**L**	**N**	**127**	**128**	**2**	**Y**	**ANRC, Census Code, Left Side**
ANRCR	**L**	**N**	**129**	**130**	**2**	**Y**	**ANRC Census Code, Right Side,**
STATEL	**L**	**N**	**131**	**132**	**2**	**Y**	**FIPS State Code, Left Side**

STATER	L	N	133	134	2	Y	FIPS State Code, Right Side
COUNTYL	L	N	135	137	3	Y	FIPS County Code, Left Side
COUNTYR	L	N	138	140	3	Y	FIPS County Code, Right Side
FMCDL	L	N	141	145	5	Y	County Subdivision FIPS PUB 55-3 Code, Left Side
FMCDR	L	N	146	150	5	Y	County Subdivision FIPS PUB 55-3 Code, Right Side
FSMCDL	L	N	151	155	5	Y	Sub-MCD FIPS PUB 55-3 Code, Left Side
FSMCDR	L	N	156	160	5	Y	Sub-MCD FIPS PUB 55-3 Code, Right Side
FPLL	L	N	161	165	5	Y	Place FIPS PUB 55-3 Code, Left Side
FPLR	L	N	166	170	5	Y	Place FIPS PUB 55-3 Code, Right Side
CTBNAL	L	N	171	176	6	Y	Census Tract/BNA Code, Left Side
	L	N	171	174	4	Y	Basic number
	L	N	175	176	2	Y	suffix
CTBNAR	L	N	177	182	6	Y	Census Tract/BNA Code, Right Side
	L	N	177	180	4	Y	Basic number
	L	N	181	182	2	Y	suffix
BLKL	L	A	183	186	4	Y	Block Number, Left Side
	L	N	183	185	3	Y	Basic number
	L	A	186	186	1	Y	suffix
BLKR	L	A	187	190	4	Y	Block Number, Right Side
	L	N	187	189	3	Y	Basic number
	L	A	190	190	1	Y	suffix
FRLONG	R	N	191	200	10	N	Start Node Longitude (Implied 6 decimal places)
FRLAT	R	N	201	209	9	N	Start Node Latitude (Implied 6 decimal places)
TOLONG	R	N	210	219	10	N	End Node Longitude (Implied 6 decimal places)
TOLAT	R	N	220	228	9	N	End Node Latitude (Implied 6 decimal places)

Type:

L = Left-Justified (numeric fields have leading zeros and may be interpreted as character data)

R = Right-justified (numeric fields do not have leading zeros, and may be interpreted as integer data)

Fmt:

A = Alphanumeric
N = Numeric

BV:

Y = Blank value is valid
N = Blank value is not valid

Record Type 2 -- Shape Point Coordinates

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "2")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
TLID	R	N	6	15	10	N	TIGER/Line™ Record ID Number (Permanent complete chain identification number)
RTSQ	R	N	16	18	3	N	Record Sequence Number (Sequentially numbered from 1 for each TLID)
LONG1	R	N	19	28	10	N	Point 1, Longitude
LAT1	R	N	29	37	9	N	Point 1, Latitude
LONG2	R	N	38	47	10	Y	Point 2, Longitude
LAT2	R	N	48	56	9	Y	Point 2, Latitude
LONG3	R	N	57	66	10	Y	Point 3, Longitude
LAT3	R	N	67	75	9	Y	Point 3, Latitude
LONG4	R	N	76	85	10	Y	Point 4, Longitude
LAT4	R	N	86	94	9	Y	Point 4, Latitude
LONG5	R	N	95	104	10	Y	Point 5, Longitude
LAT5	R	N	105	113	9	Y	Point 5, Latitude
LONG6	R	N	114	123	10	Y	Point 6, Longitude
LAT6	R	N	124	132	9	Y	Point 6, Latitude
LONG7	R	N	133	142	10	Y	Point 7, Longitude
LAT7	R	N	143	151	9	Y	Point 7, Latitude
LONG8	R	N	152	161	10	Y	Point 8, Longitude
LAT8	R	N	162	170	9	Y	Point 8, Latitude
LONG9	R	N	171	180	10	Y	Point 9, Longitude
LAT9	R	N	181	189	9	Y	Point 9, Latitude
LONG10	R	N	190	199	10	Y	Point 10, Longitude
LAT10	R	N	200	208	9	Y	Point 10, Latitude

Note: **The TIGER/Line™ files contain a maximum of 10 shape points on one record. The number of shape point records for a complete chain may be 0, 1, or more. Coordinates have an implied 6 decimal places.**

Record Type 3 -- Additional 1990 and 1980 Decennial Census Geographic Entity Codes

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "3")
VERSION	L	N	2	5	4	N	Version Number (Value "0005" identifies the 1992 TIGER/Line files)
TLID	R	N	6	15	10	N	TIGER/Line™ Record ID Number (Permanent complete chain identification number)
STATE80L	L	N	16	17	2	Y	1980 FIPS State Code, Left Side
STATE80R	L	N	18	19	2	Y	1980 FIPS State Code, Right Side
COUN80L	L	N	20	22	3	Y	1980 FIPS County Code, Left Side
COUN80R	L	N	23	25	3	Y	1980 FIPS County Code, Right Side
FMCD80L	L	N	26	30	5	Y	County Subdivision, 1980 FIPS PUB 55-3 Code, Left Side
FMCD80R	L	N	31	35	5	Y	County Subdivision, 1980 FIPS PUB 55-3 Code, Right Side
FPL80L	L	N	36	40	5	Y	Place, 1980 FIPS PUB 55-3 Code, Left Side
FPL80R	L	N	41	45	5	Y	Place, 1980 FIPS PUB 55-3 Code, Right Side
CTBNA80L	L	N	46	51	6	Y	1980 Census Tract/BNA Code, Left Side
	L	N	46	49	4	Y	Basic number
	L	N	50	51	2	Y	suffix
CTBNA80R	L	N	52	57	6	Y	1980 Census Tract/BNA Code, Right Side
	L	N	52	55	4	Y	Basic number
	L	N	56	57	2	Y	suffix
BLK80L	L	N	58	60	3	Y	1980 Block Number, Left Side
BLK80R	L	N	61	63	3	Y	1980 Block Number, Right Side
MCD80L	L	N	64	66	3	Y	County Subdivision, 1980 Census Code, Left Side
MCD80R	L	N	67	69	3	Y	County Subdivision, 1980 Census Code, Right Side
PL80L	L	N	70	73	4	Y	Place, 1980 Census Code, Left Side
PL80R	L	N	74	77	4	Y	Place, 1980 Census Code, Right Side
AIRL	L	N	78	81	4	Y	AI/ANA Census Code, Left Side
AIRR	L	N	82	85	4	Y	AI/ANA Census Code, Right Side
MCDL	L	N	86	88	3	Y	County Subdivision Census

							Code, Left Side
MCDR	L	N	89	91	3	Y	County Subdivision Census Code, Right Side
SMCDL	L	N	92	93	2	Y	Sub-MCD Census Code, Left Side
SMCDR	L	N	94	95	2	Y	Sub-MCD Census Code, Right Side
PLL	L	N	96	99	4	Y	Place, Census Code, Left Side
PLR	L	N	100	103	4	Y	Place, Census Code, Right Side
VTDL	L	A	104	107	4	Y	1990 VTD Code, Left Side
VTDR	L	A	108	111	4	Y	1990 VTD Code, Right Side

Record Type 4 -- Index to Alternate Feature Identifiers

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "4")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
TLID	R	N	6	15	10	N	TIGER/Line™ Record ID Number (Permanent complete chain identification numbers)
RTSQ	R	N	16	18	3	N	Record Sequence Number (Sequentially numbered from 1 for each TLID/Complete Chain)
FEAT1	R	N	19	26	8	N	Identification Number for 1st Alternate Feature Identifier
FEAT2	R	N	27	34	8	Y	Identification Number for 2nd Alternate Feature Identifier
FEAT3	R	N	35	42	8	Y	Identification Number for 3rd Alternate Feature Identifier
FEAT4	R	N	43	50	8	Y	Identification Number for 4th Alternate Feature Identifier
FEAT5	R	N	51	58	8	Y	Identification Number for 5th Alternate Feature Identifier

Record Type 5 -- Feature Identifier List

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "5")
STATE	L	N	2	3	2	N	FIPS State Code
COUNTY	L	N	4	6	3	N	FIPS County Code
FEAT	R	N	7	14	8	N	Identification Number for the Feature Identifier
FEDIRP	L	A	15	16	2	Y	Feature Direction, Prefix
FENAME	L	A	17	46	30	Y	Feature Name
FETYPE	L	A	47	50	4	Y	Feature Type
FEDIRS	L	A	51	52	2	Y	Feature Direction, Suffix

Record Type 6 -- Additional Address Range and ZIP Code ®Data

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is 6")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
TLID	R	N	6	15	10	N	TIGER/Line™ Record ID Number (Permanent complete chain identification numbers)
RTSQ	R	N	16	18	3	N	Record Sequence Number (Sequentially numbered from 1 for each TLID)
FRADDL	R	A	19	29	11	Y	Start Address, Left Side
TOADDL	R	A	30	40	11	Y	End Address, Left Side
FRADDR	R	A	41	51	11	Y	Start Address,Right Side
TOADDR	R	A	52	62	11	Y	End Address, Right Side
FRIADDL	R	N	63	63	1	Y	Start Address, Impute Flag, Left Side
TOIADDL	R	N	64	64	1	Y	End Address, Impute Flag, Left Side
FRIADDR	R	N	65	65	1	Y	Start Address, Impute Flag, Right Side
TOIADDR	R	N	66	66	1	Y	End Address, Impute Flag, Right Side
ZIPL	L	N	67	71	5	Y	ZIP Code®, Left Side (A non-blank value appears only when left address range is present)
ZIPR	L	N	72	76	5	Y	ZIP Code®, Right Side (A non-blank value appears only when left address range is present)

Record Type 7 -- Landmark Features

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "7")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
STATE	L	N	6	7	2	N	FIPS State Code
COUNTY	L	N	8	10	3	N	FIPS County Code
LAND	R	N	11	20	10	N	Landmark Identification Number
SOURCE	L	A	21	21	1	Y	Source Code
CFCC	L	A	22	24	3	Y	CFCC
LANAME	L	A	25	54	30	Y	Landmark Feature Identifier
LALONG	R	N	55	64	10	Y	Longitude (Implied 6 decimal places, only for point landmarks)
LALAT	R	N	65	73	9	Y	Latitude (Implied 6 decimal places, only for point landmarks)
FILLER	L	A	74	74	1	Y	Filler (to make even character count) (contains a blank character space)

Record Type 8 -- Polygons Linked to Area Landmarks

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "8")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
STATE	L	N	6	7	2	N	FIPS State Code
COUNTY	L	N	8	10	3	N	FIPS County Code
CENID	L	N	11	15	5	N	Census File Identification Code
POLYID	R	N	16	25	10	N	Polygon Identification Number (Polygon number is unique to CENID)
LAND	R	N	26	35	10	N	Landmark Identification Number
FILLER	L	A	36	36	1	Y	Filler (to make even character count) (contains a blank character space)

Record Type A—Additional Polygon Geographic Entity Codes

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "A")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
STATE	L	N	6	7	2	N	FIPS State Code
COUNTY	L	N	8	10	3	N	FIPS County Code
CENID	L	N	11	15	5	N	Census File Identification Code
POLYID	R	N	16	25	10	N	Polygon Identification number (number is unique to CENID)
FAIR	L	N	26	30	5	Y	AI/ANA FIPS PUB 55-3 Code
FMCD	L	N	31	35	5	N	County Subdivision FIPS PUB 55-3 Code
FPL	L	N	36	40	5	Y	Place FIPS PUB 55-3 Code
CTBNA	L	N	41	46	6	N	Census Tract/BNA Code
	L	N	41	44	4	N	Basic number
	L	N	45	46	2	Y	suffix
BLK	L	A	47	50	4	N	Block Number
	L	N	47	49	3	N	Basic number
	L	A	50	50	1	Y	suffix
CD101	L	N	51	52	2	Y	101st Congressional District Code
CD103	L	N	53	54	2	Y	103rd Congressional District Code
SDELM	L	A	55	59	5	Y	Elementary School District Code
SDMID	L	A	60	64	5	Y	Middle School District Code
SDSEC	L	A	65	69	5	Y	Secondary School District Code
SDUNI	L	A	70	74	5	Y	Unified School District Code
TAZ	L	A	75	80	6	Y	TAZ Code
UA	L	N	81	84	4	Y	Census UA Code
URBFLAG	L	A	85	85	1	N	U/R Flag
RS	L	A	86	98	13	Y	Reserved Space (The field is reserved, but currently contains a blank character space)

Record Type F—Corrected Geographic Area Codes for the 1990 Census*

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "F")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
STATE	L	N	6	7	2	N	FIPS State Code
COUNTY	L	N	8	10	3	N	FIPS County Code
CENID	L	N	11	15	5	N	Census File Identification Code
POLYID	R	N	16	25	10	N	Polygon Identification Number (number is unique to CENID)
STATE90	L	N	26	27	2	N	1990 FIPS State Code
COUNTY90	L	N	28	30	3	N	1990 FIPS County Code
FAIR90	L	N	31	35	5	Y	1990 AI/ANA FIPS PUB 55-3 Code
FMCD90	L	N	36	40	5	Y	1990 County Subdivision FIPS PUB 55-3 Code
FSMCD90	L	N	41	45	5	Y	1990 Sub-MCD FIPS PUB 55-3 Code
FPL90	L	N	46	50	5	Y	1990 Place FIPS PUB 55-3 Code
CTBNA90	L	N	51	56	6	Y	1990 Census Tract/BNA Code
	L	N	51	54	4	Y	Basic number
	L	N	55	56	2	Y	suffix
BLK90	L	A	57	61	5	Y	1990 Block Number
	L	N	57	59	3	Y	Basic number
	L	A	60	60	1	Y	2 character suffix
	L	A	61	61	1	Y	Collection Suffix
FILLER	L	A	62	62	1	Y	Filler (to make even character count) (contains a blank character space)

***Present only when different from Record Type 1 or A.**

Record Type G -- 1992 Geographic Codes and Entity Changes*

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	**L**	**A**	**1**	**1**	**1**	**N**	**Record Type (Record Type is "G")**
VERSION	**L**	**N**	**2**	**5**	**4**	**N**	**Version Number (Value "0005" identifies the 1992 TIGER/Line files)**
STATE	**L**	**N**	**6**	**7**	**2**	**N**	**FIPS State Code**
COUNTY	**L**	**N**	**8**	**10**	**3**	**N**	**FIPS County Code**
CENID	**L**	**N**	**11**	**15**	**5**	**N**	**Census File Identification Code**
POLYID	**R**	**N**	**16**	**25**	**10**	**N**	**Polygon Identification Number (number is unique to CENID)**
STATECU	**L**	**N**	**26**	**27**	**2**	**N**	**Current FIPS State Code**
COUNTYCU	**L**	**N**	**28**	**30**	**3**	**N**	**Current FIPS County Code**
FAIRCU	**L**	**N**	**31**	**35**	**5**	**Y**	**Current AI/ANA FIPS PUB 55-3 Code**
FMCDCU	**L**	**N**	**36**	**40**	**5**	**Y**	**Current County Subdivision FIPS PUB 55-3 Code**
FSMCDCU	**L**	**N**	**41**	**45**	**5**	**Y**	**Current Sub-MCD FIPS PUB 55-3 Code**
FPLCU	**L**	**N**	**46**	**50**	**5**	**Y**	**Current Place FIPS PUB 55-3 Code**
CDCU	**L**	**N**	**51**	**52**	**2**	**Y**	**Current 103rd Congressional District Code**

***Present only when different from Record Type 1 or A.**

Record Type I: The Link Between Complete Chains and Polygons

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "I")
VERSION	L	N	2	5	4	N	Version Number (Value "0005" identifies the 1992 TIGER/Line files)
TLID	R	N	6	15	10	N	TIGER/Line™ Record ID Number (Permanent complete chain identification number)
STATE	L	N	16	17	2	N	FIPS State Code
COUNTY	L	N	18	20	3	N	FIPS County Code
RTLINK	L	A	21	21	1	N	Area Pointer Type Code ("P" = polygon Identification code)
CENIDL	L	N	22	26	5	Y	Census File Identification Code, Left Side
POLYIDL	R	N	27	36	10	Y	Polygon Identification Number, Left Side (number is unique to CENID)
CENIDR	L	N	37	41	5	Y	Census File Identification Code, Right Side
POLYIDR	R	N	42	51	10	Y	Polygon Identification Number, Right Side
FILLER	L	A	52	52	1	Y	Filler (to make even character count) (contains a blank character space)

Record Type P—Polygon Internal Point

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "P")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
STATE	L	N	6	7	2	N	FIPS State Code
COUNTY	L	N	8	10	3	N	FIPS County Code
CENID	L	N	11	15	5	N	Census File Identification Code
POLYID	R	N	16	25	10	N	Polygon Identification Number, unique to CENID
POLYLONG	R	N	26	35	10	N	Longitude (Implied 6 decimal places)
POLYLAT	R	N	36	44	9	N	Latitude (Implied 6 decimal places)

Record Type R—Record Number Range

Field	Type	Fmt	Beg	End	Size	BV	Description and Notes
RT	L	A	1	1	1	N	Record Type (Record Type is "R")
VERSION	L	N	2	5	4	N	Version (Value "0005" identifies the 1992 TIGER/Line files)
STATE	L	N	6	7	2	N	FIPS State Code
COUNTY	L	N	8	10	3	N	FIPS County Code
CENID	L	N	11	15	5	N	Census File Identification Code
MAXID	R	N	16	25	10	N	Maximum TLID Value for this CENID (For all TIGER/Line™ files using this CENID)
MINID	R	N	26	35	10	N	Minimum TLID Value for this CENID (For all TIGER/Line™ files using this CENID)
HIGHID	R	N	36	45	10	N	Current TLID Value (Used for this CENID in this file version)
FILLER	L	A	46	46	1	N	Filler (to make even character count) (contains a blank character space)

Note: See Appendix B for a list of field name changes since the 1990 Census TIGER/Line files.

Type: L = Left-Justified (numeric fields have leading zeros and may be interpreted as character data)
R = Right-justified (numeric fields do not have leading zeros, and may be interpreted as integer data)

Fmt: A = Alphanumeric
N = Numeric

BV: Y = Blank value is valid
N = Blank value is not valid

APPENDIX C: 104th

DESCRIPTION

Physical Description

The collection was received as 1 CD-ROM with the label "House of Representatives, 104th Congress, TEXT (version 2-19-98), Office of the Clerk". The CD_ROM contains 11,437 ASCII text (MS-DOS) files occupying 317 MB.

Information model

All files in the collection follow the naming convention
filename = <first><number><second>.txt
where <first> is one of the prefixes in [h hc hj hr s sc sj sr]

It is our understanding that the two letters of the prefix represent the following information about the origination of the specific document:

Prefix	Meaning
h	House of Representatives
hc	House of Representatives concurrent resolution
hj	House of Representatives joint resolution
hr	House of Representatives resolution
s	Senate
sc	Senate concurrent resolution
sj	Senate joint resolution
sr	Senate resolution

Table C-1. Document origin for the 104th collection

The <number> in the file name is the actual number of the resolution or bill or act.

The <second> part of the file name is a suffix from the list: [as ath ats cdh cds cph cps eah eas eh enr es hds ih ips is lth pch pcs pp rch rcs rds rfh rfs rh rh2 rih ris rs rs2 rth rts sc]. The meaning of each suffix is not entirely clear. However, we believe that the suffixes refer to the place (House or Senate) where an action is being taken on the document, and encodes the following meta-data:

Suffix	Count	Meaning
as	3	amendment in the Senate
ath	97	considered and agreed to in the House
ats	251	considered and agreed to in the Senate

cdh	6	considered in the House (and committed to a committee)
cds	1	considered in the Senate (and committed to a committee)
cph	19	considered and passed in the House
cps	20	considered and passed in the Senate
eah	36	resolved with amendments in the House
eas	115	resolved with amendments in the Senate
eh	887	enactment resolved in the House
es	270	enactment resolved in the Senate
enr	413	
hds	3	held at the desk in the Senate
ih	4947	introduced in the House
is	2271	introduced in the Senate
ips	4	indefinitely postponed in the Senate
lth	8	laid on the table in the House
pch	1	placed on calendar in the House
pcs	184	placed on calendar in the Senate
pp	31	ordered to be printed as passed
rch	8	re-referred to the same or a different committee in the House
rcs	6	re-referred to the same or a different committee in the Senate
rds	187	received in the Senate
rfh	103	referred to a committee in the House
rih	1	referred to a committee for a limited time in the House
ris	44	referred to a committee for a limited time to report, or be discharged and placed on the calendar in the Senate
rh	742	report in the House
rs	471	report in the Senate
rh2	1	2-part report in the House
rs2	3	2-part report in the Senate
rth	1	referred to committee in the House
rts	5	referred to committee in the Senate
sc	1	

Table C-2. Data Dictionary for the 104th

There appears to be some ambiguities in the naming convention. For example, the distinction between the suffixes “rch” and “rth” is unclear. The suffix “sc” appears to be inconsistent with the generally adopted convention of encoding where the document is being acted upon. We strongly suggest that for a collection of the proceedings of the Congress, such encoding of meta-data be made explicitly available.

APPENDIX D: VAD97

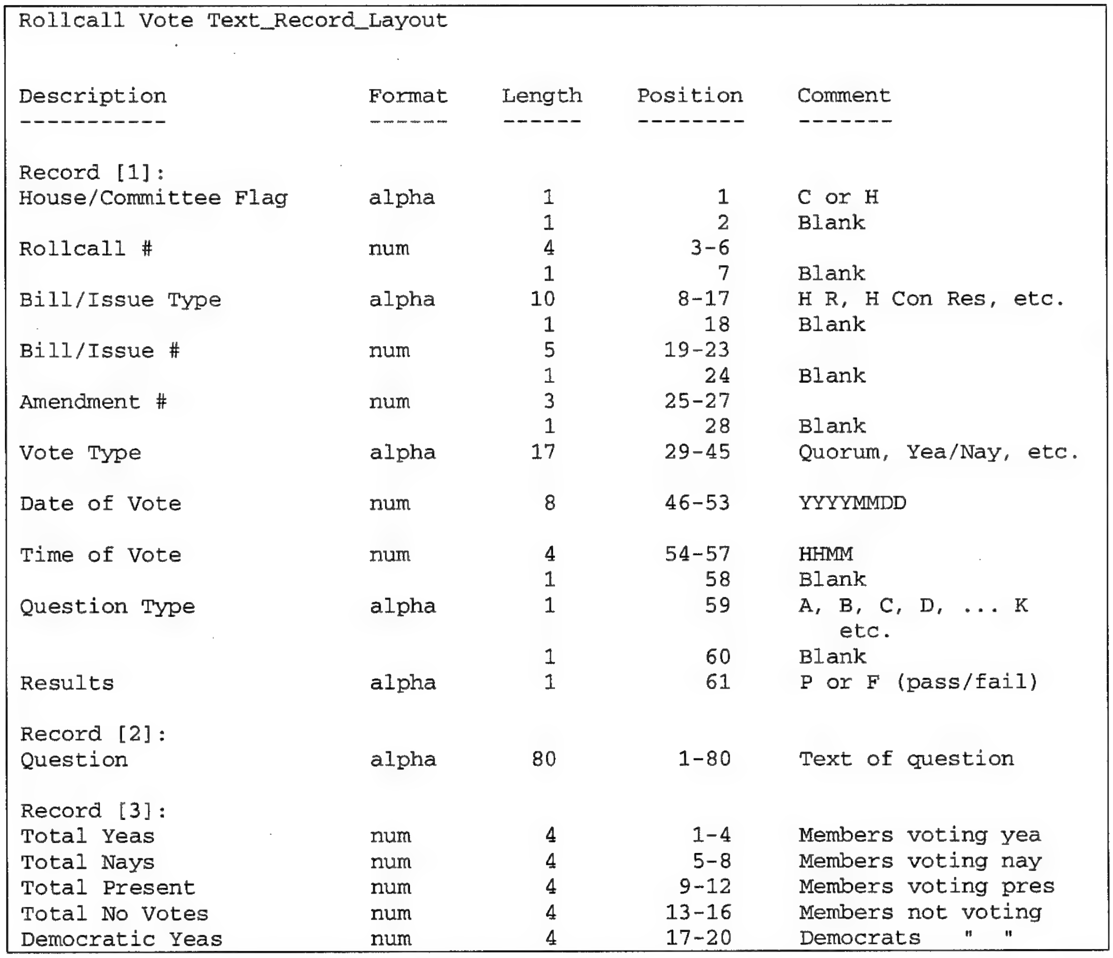
PHYSICAL SOURCE:
1 CD (33 MB) containing 640 digital objects

COLLECTION LEVEL STRUCTURE/META-DATA:
Main data: *roll001.txt, ..., roll640.txt* (640 files) each containing the data for exactly one roll call vote

The exact format of this data is described in the file:
Vote_Ascii_Text_Record_Layout.txt

Additionally: HTML files (*index.htm, ROLL_000.htm, ROLL_100.htm, ..., ROLL_600.htm* (7 files), *vote001.htm, ..., vote640.htm* (640 files))

OBJECT LEVEL STRUCTURE/META-DATA:
Each object is a roll call vote and is highly structured (format: see *Vote_Ascii_Text_Record_Layout.txt*)



```
Rollcall Vote Text_Record_Layout


Description               Format   Length   Position    Comment
-----------               ------   ------   --------    -------

Record [1]:
House/Committee Flag      alpha      1           1      C or H
                                     1           2      Blank
Rollcall #                num        4         3-6
                                     1           7      Blank
Bill/Issue Type           alpha     10        8-17      H R, H Con Res, etc.
                                     1          18      Blank
Bill/Issue #              num        5       19-23
                                     1          24      Blank
Amendment #               num        3       25-27
                                     1          28      Blank
Vote Type                 alpha     17       29-45      Quorum, Yea/Nay, etc.

Date of Vote              num        8       46-53      YYYYMMDD

Time of Vote              num        4       54-57      HHMM
                                     1          58      Blank
Question Type             alpha      1          59      A, B, C, D, ... K
                                                           etc.
                                     1          60      Blank
Results                   alpha      1          61      P or F (pass/fail)

Record [2]:
Question                  alpha     80        1-80      Text of question

Record [3]:
Total Yeas                num        4         1-4      Members voting yea
Total Nays                num        4         5-8      Members voting nay
Total Present             num        4        9-12      Members voting pres
Total No Votes            num        4       13-16      Members not voting
Democratic Yeas           num        4       17-20      Democrats   "   "
Democratic Nays          num        4      21-24     Democrats   "   "
Democratic Presents      num        4      25-28     Democrats   "   "
Democratic No Votes      num        4      29-32     Democrats   "   "
Republican Yeas          num        4      33-36     Republican  "   "
Republican Nays          num        4      37-40     Republican  "   "
Republican Presents      num        4      41-44     Republican  "   "
Republican No Votes      num        4      45-48     Republican  "   "
Independent Yeas         num        4      49-52     Independent "   "
Independent Nays         num        4      53-56     Independent "   "
Independent Presents     num        4      57-60     Independent "   "
Independent No Votes     num        4      61-64     Independent "   "

Description              Format   Length   Position   Comment
-----------              ------   ------   --------   -------

Records [4] thru [# members voting]:
Member Name              alpha     20        1-20     Name
                                    1          21     Blank
Archive #                num        8       22-29     yr/mo/state/dist
                                    1          30     Blank
Party                    alpha      1          31     D, R, or I
                                    1          32     Blank
Vote Preference          alpha      4       33-36     "YEA ", "NAY ",
                                                      "PRES", "NV  "
                                    1          37     Blank
State Digraph            alpha      2          38     "VA" etc.. or "XX"
                                    1          40     Blank
State Name               alpha     15          41     "VIRGINIA" etc..
                                                        or "DELEGATES"

Note: Each record will be 82 chars long with cr/lf in position 81/82...
      Numeric data is right justified, blank filled.
```

APPENDIX E: EAP

PHYSICAL SOURCE:

- 1 DAT tape 4mm containing 400 .tiff images (batch #1),
- 2 CDs containing *.jpg* and *.gif* images (913 MB)
- 1 floppy disk (1.4MB) with a NAIL lookup table (5MB uncompressed)

COLLECTION LEVEL STRUCTURE / META-DATA:

The EAP objects (images) are grouped into:

- batches: **EAP\1 ... EAP\5**

where each batch contains

- up to 11 subdirectories, e.g. **EAP\4\1 ... EAP\4\11**

where each of these subdirectories contains

- up to 200 files

It seems that the organization into batches and subdirectories within the batches doesn't carry semantic information.

The filename XX-YYYYZ.EXT allows us to reconstruct some information:

• XX:	batch number
• YYYY:	number within batch
• Z:	is T or A ("thumbnail"? vs "all"?)
• EXT:	GIF/JPG

APPENDIX F: Vietnam

PHYSICAL SOURCE:

- 5 class 3480 tape cartridges (volume IDs: 007726, 008817, 009928, 5021, 2431) containing ...
- 8 versions of the CACCF database, encoded in EBCDIC, and one hardcopy documentation describing the record structure. In order to understand all encodings, the corresponding DoD Instruction (7730.22 dated March 20, 1973) has to be archived as well!

Each CACCF file is a set of fixed-length data records, one record for each casualty. Due to the highly regular structure, it is straightforward to design an XML DTD or a relational database schema for this collection.

The structure of the test corpus is as follows: The *CACCF collection* consists of 8 different versions of the *CACCF databases*. Each of these versions is identifiable by a creation date, which is apparently encoded into the filename of the source files as follows:

RG330.CAC.<X><YY><MM><DD>

Here, X is either "P" or "C" (meaning unclear) and YYMMDD is the date.

APPENDIX G: AMICO

PHYSICAL SOURCE

The sample AMICO collection consists of about 50 images, including thumbnails (in *jpg*) and full images (in *tif*), and about 70 meta-data records describing the art objects as well as the digital image objects, with each record having about 150 attributes. The total size of the collection sample is about 125MB.

COLLECTION LEVEL STRUCTURE/META-DATA

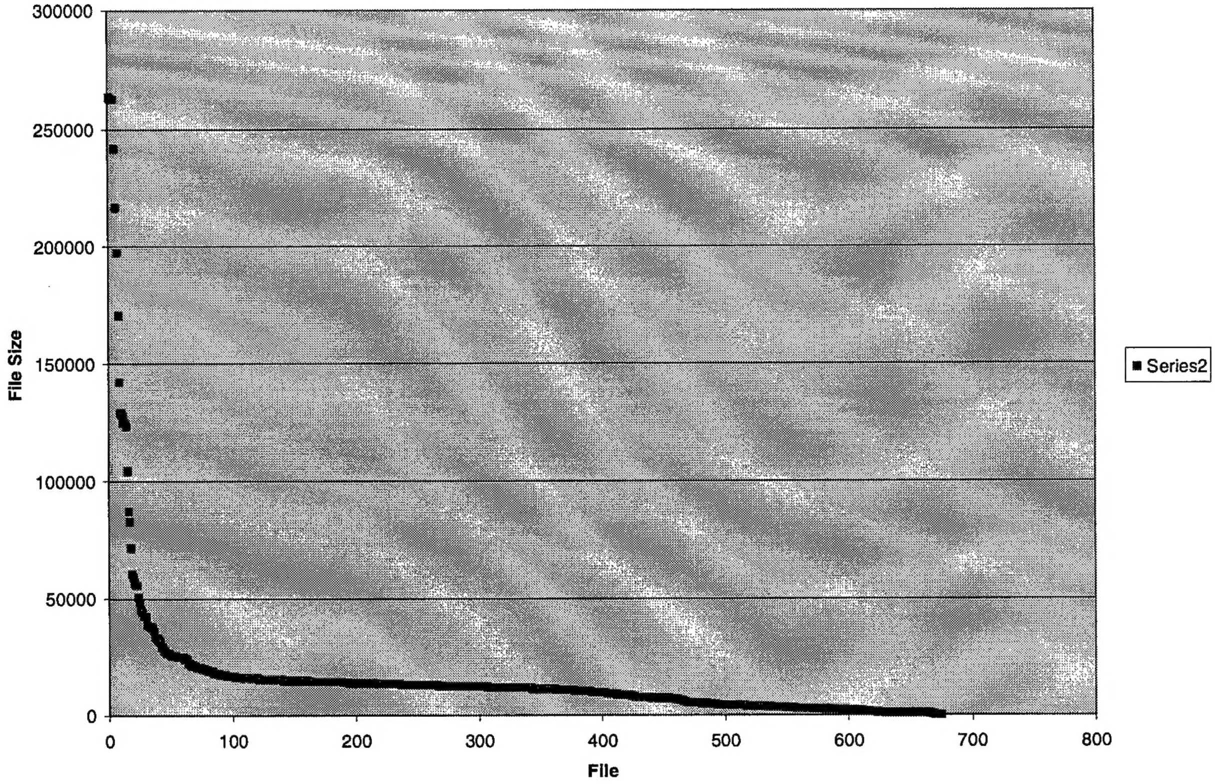
The procedures applied to the sample collection are exactly the same as the ones that will be applied to the complete AMICO collection that will be made available to SDSC. The image data files will be archived in HPSS via the SDSC Storage Resource Broker (SRB) while the meta-data will be archived as searchable XML documents.

APPENDIX H: JTIC

PHYSICAL SOURCE

1 CD, 680 files, 388 MB.

Figure1. File Size Distribution



The above collections of files has a skewed size distribution. Of the total size of 388MB, a single file accounts for 297MB. The next two largest files account for about 75MB (50MB+25MB), and the subsequent two largest files account for about 5MB (4MB+1MB). The remaining 675 files account for the remaining 10MB. Figure 1 shows the file size distribution for the remaining files. These sizes range from 24bytes to 264Kbytes.

Based on the analysis of file size distribution, the set of files can be grouped into three clusters: less than 100KB, 100KB to 5M, and 5M to 500M. This can be a useful classification scheme for archival storage purposes.

All the files associated with the JITC collection appear in a single directory. The input does not contain an inherent directory structure. More than 80% of the files have a file name *extension* (typically a 3-letter extension) as part of the file name (e.g., *doc, jpg*).

This extension can be used to classify files in the collection. Each collection can be viewed as consisting of one or more sub-collections, where each sub-collection is a set of files having the same file extension. Files that do not have such an extension will have to be grouped together. In the JITC collection there are 114 such files that have no extension.